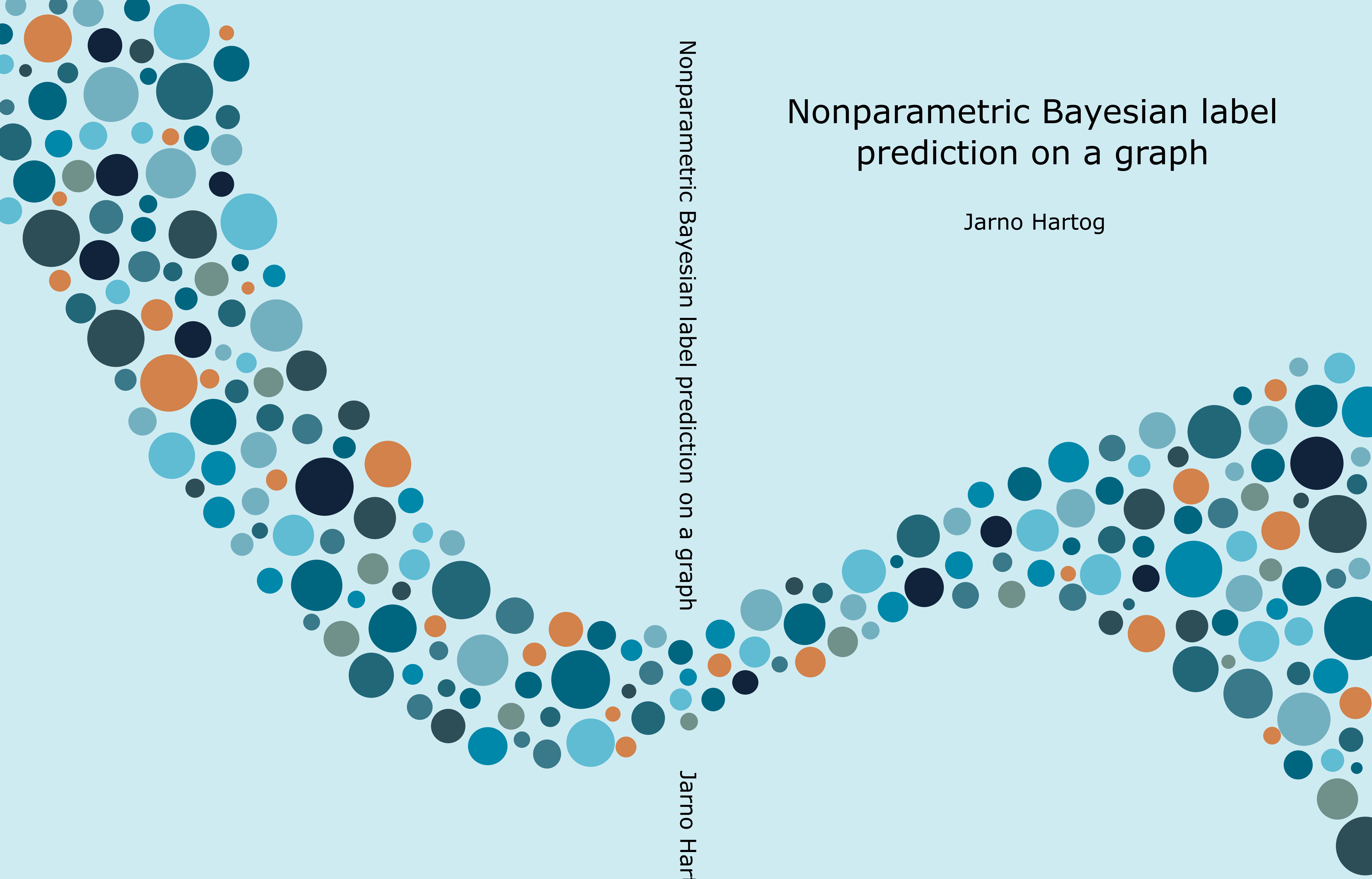
Nonparametric Bayesian label prediction on a graph

Hartog, J.



Nonparametric Bayesian label prediction on a graph

Jarno Hartog

Nonparametric Bayesian label prediction on a graph

Jarno Hartog

NONPARAMETRIC BAYESIAN LABEL PREDICTION ON A GRAPH

NONPARAMETRIC BAYESIAN LABEL PREDICTION ON A GRAPH

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op woensdag 11 september 2019, te 14.00 uur

door

Jarno HARTOG

geboren te Vlaardingen

UNIVERSITY OF AMSTERDAM

NWO
Nederlandse Organisatie voor Wetenschappelijk Onderzoek

Keywords: binary classification, graph, Bayesian nonparametrics

Printed by: Ipskamp Printing

Front & Back: M.I. Stout & J. Hartog

ISBN 978-04-028-1616-7

An electronic version of this dissertation is available at
`https://dare.uva.nl/`

Voor Merel Isabel Stout

CONTENTS

SUMMARY

This dissertation is devoted to nonparametric Bayesian label prediction on a graph. Label prediction is a problem within the area of statistical learning. Typically, we want to predict a label, i.e. the outcome of some quantitative or categorical measurement, based on some features of the available data. A training set of observed, possibly noisy, labels and the corresponding features is available to base our inference on.

Label prediction problems on graphs arise in a variety of applications, for instance in machine learning, in the prediction of the biological function of a protein in a protein-protein interaction graph, in image analysis, and in the prediction of brand preference in social networks.

Prime examples are problems in which the graph is given by the application context. In our setting, we think of the labels as a stochastic process indexed by the graph vertex set. The data are noisy observations of the vertex labels. In case of regression on the graph, this process is real-valued. For classification problems, the process can be binary-valued or discrete-valued in general. Our main goal is to predict the missing labels correctly. The underlying idea is that the location of a given vertex in the graph and the information of its neighbors should have predictive power for the label of the vertex of interest.

First, we consider a hierarchical Bayesian approach with a randomly scaled Gaussian prior. A benefit of the hierarchical Bayesian procedure is that it is capable, when properly designed, to automatically determine the correct value for the scale parameter. The prior can be interpreted as a series expansion of the function of interest using the eigenvectors of the graph Laplacian matrix as a basis. In order to alleviate the computational burden of taking into account all the eigenvectors, we truncate the series at a random point to make our method applicable for very large graphs. For the full Bayesian approach, we sample from the joint posterior of the function of interest, regularization parameter and truncation level. Alternatively, we use empirical Bayes methods to select the regularization parameter and truncation level. After fixing these parameters, we sample from the posterior distribution of the function of interest. Finally, we use variational inference to approximate the posterior distribution. A major benefit of variational inference is that it allows for stochastic optimization to make it applicable to large data.

Samenvatting

Dit proefschrift is gewijd aan het op een nietparametrische Bayesiaanse wijze voorspellen van labels op een graaf. Labelvoorspelling is een statistisch leerprobleem waar we de uitkomst van een kwantitatief of categorisch experiment willen voorspellen op basis van de beschikbare data. Een trainingsdataset van geobserveerde labels, mogelijk met ruis, en de corresponderende kenmerken zijn beschikbaar om onze inferentie op te baseren.

Labelvoorspellingsproblemen duiken in verschillende toepassingsgebieden op, bijvoorbeeld in automatisch leren, in het voorspellen van de biologische functie van een proteïne in een proteïne-proteïne interactiegraaf, in beeldanalyse en in het voorspellen van merkvoorkeuren in een sociaal netwerk. De meest relevante problemen zijn die waarin de graaf is gegeven door de context van de toepassing. De beschikbare data kan ruis vertonen en bestaat uit observaties van labels op de knooppunten van een graaf. Ons hoofddoel is het correct voorspellen van de missende labels. In onze opzet modelleren we de labels als een stochastisch proces geïndexeerd door de knooppunten. De onderliggende gedachte is dat de locatie van een gegeven knooppunt en de informatie van zijn buren voorspellende kracht heeft voor het label van het betreffende knooppunt.

In het geval van regressie op de graaf neemt dit proces reeële waarden aan en in classificatieproblemen discrete waarden. Binnen dit onderzoek focussen we in het bijzonder op classificatieproblemen met binaire waarden. We beginnen met een hiërarchische Bayesiaanse aanpak met een willekeurig geschaalde Gaussische a-priori-verdeling. Indien goed ontworpen, is de hiërarchische Bayesiaanse procedure in staat om de juiste schaalparameterwaarde automatisch te vinden. De a-priori-verdeling kan worden geïnterpreteerd als een reeksontwikkeling van de onderliggende functie over een basis van eigenvectoren van de Laplaciaanse matrix van de graaf. Om de computationele last van het meenemen van alle eigenvectoren te verlichten, kappen we de reeks op een willekeurig moment af om onze methode toepasbaar te maken voor grote grafen.

Voor de volledige Bayesiaanse aanpak nemen we een steekproef uit de gezamenlijke a-posteriori-verdeling van de onderliggende functie, de regularisatieparameter en het afkapniveau. Daarnaast kijken we ook naar een empirisch Bayesiaanse aanpak om afzonderlijk de regularisatieparameter en het afkapniveau te selecteren. Nadat we deze parameters hebben vastgesteld, nemen we een steekproef van de a-posteriori-verdeling van de onderliggende functie. Tot slot gebruiken we ook variationele methoden om de a-posteriori-verdeling te benaderen. Variationele methoden kunnen we met behulp van stochastische optimalisatie op zeer grote datasets toepassen.

1

INTRODUCTION

This dissertation is devoted to nonparametric Bayesian label prediction on a graph. In this chapter, we give an introduction to our main subject, preliminary definitions and results that we need in our work. We introduce key concepts from graph theory such as the graph Laplacian matrix and a notion of smoothness for functions on the vertices of the graph. We briefly introduce Bayesian statistics and Markov chain Monte Carlo methods for the computation of the posterior distribution. Finally, we introduce label prediction problems on a graph and outline the rest of this dissertation.

Label prediction is a problem within the area of statistical learning. Typically, we want to predict a *label*, i.e. the outcome of a quantitative or categorical measurement based on *features* of the available data. A *training set* of observed, possibly noisy, labels and the corresponding features is available to base our inference on. Within machine learning literature, a difference is made between *supervised learning*, where both the labels and features are observed, and *unsupervised learning*, where only the features are observed, see for example Friedman et al. (2001). This last category includes clustering problems and usually exploits the presence of special structure of the data. In this dissertation, we consider *semi-supervised learning* problems where the data consists of noisy observations of some of the labels, and the data is structured as a graph.

Label prediction problems on graphs arise in a variety of applications, for instance in machine learning (e.g. Belkin et al. (2004); Sindhwani et al. (2007)), in the prediction of the biological function of a protein in a protein-protein interaction graph (e.g. Kolaczyk (2009); Nariai et al. (2007); Sharan et al. (2007)), in image analysis (e.g. Liu et al. (2014)) and in the prediction of brand preference in social networks (Blair et al. (2003); Smith (2011)).

Prime examples are problems in which the graph is given by the application context. If a graph is not given by the application, one can still construct a graph in some cases. For example, if the data are points in a metric space, a graph can be constructed by connecting data points that are close. The data are noisy observations of some of the vertex labels. In our context, we think of the labels as a stochastic process indexed by the graph vertex set. The process is real-valued in case of regression on the graph. For classification problems, the process can be binary-valued or discrete-valued in general. Our main goal is to predict the missing labels correctly. The underlying idea is that the location of a given vertex in the graph and the information of its neighbors should have predictive power for the label of the vertex of interest.

Several graph-based prediction methods exist in the literature. The simplest is *nearest neighbor* prediction. In this method an unknown label is predicted by taking the average of the observed labels in the neighborhood of the vertex of interest. The neighborhood consists of the vertices directly connected to the vertex of interest. This can also be generalized to vertices that are more than one step away. There are several options on how to take missing observations into account. Missing labels in the neighborhood of the vertex of interest can, for example, simply be ignored or be imputed. The more interesting choice, however, is the size of the neighborhood. This corresponds to how much we smooth the data over the graph. For more information about nearest neighbor methods, see Friedman et al. (2001). Other approaches are *Markov random field* models and *kernel-based* methods, see Kolaczyk (2009). We consider, in this dissertation, methods closely related to frequentist kernel-based methods. We expand on the Bayesian framework proposed in Kirichenko and Van Zanten (2017). A key similarity is the use of the *Laplacian matrix* associated to the graph to take the graph geometry into account (cf. Ando and Zhang (2007); Belkin et al. (2004); Kolaczyk (2009); Zhu and Hastie (2005)). Regularization using a different norm than the one induced by the graph Laplacian matrix is discussed in Sadhanala et al. (2016) and a different Bayesian approach is presented in Bertozzi et al. (2018).

The rest of this dissertation is organized as follows. In the remainder of this chapter, we introduce preliminary concepts from graph theory, statistics and Bayesian computation to formalize our problem setting.

CHAPTER 2

In Chapter 2, we describe the implementation of a nonparametric Bayesian approach to solve regression and binary classification problems on graphs. We consider a hierarchical Bayesian approach with a randomly scaled Gaussian prior. A benefit of the hierarchical Bayesian procedure is that it is capable, when properly designed, to automatically determine the correct value for the regularization parameter. The prior distribution uses the graph Laplacian to take the underlying geometry of the graph into account. We propose two methods. The first is based on a theoretically optimal prior. The second is a more flexible variant using partial conjugacy. In order to illustrate the proposed methods we have two simulated data examples and two examples using real data.

CHAPTER 3

We improve the method from Chapter 2 in Chapter 3. The prior in Chapter 2 can be interpreted as a series expansion of the *soft label function* using the eigenvectors of the graph Laplacian matrix as a basis. The coefficients have a Gaussian distribution and the series is randomly scaled with a regularization parameter. Using all eigenvectors is computationally demanding and limits the applicability of our method for very large graphs. In order to alleviate this issue we truncate the series at a random in Chapter 3 for computational efficiency. Truncating the series at a random point also makes the prior more flexible in terms of adaptation to smoothness. We compare our truncated prior to the untruncated prior in simulated and real data examples to illustrate the improved scalability in terms of the size of the underlying graph.

CHAPTER 4
In Chapter 4, we use *empirical Bayes* methods to select the regularization parameter and truncation level. This is a different approach than the *Markov chain Monte Carlo* algorithm used to sample from the posterior distribution in Chapters 2 and 3. In the MCMC algorithm, the regularization parameter and truncation level are simulated simultaneously with the parameter of interest. In the empirical Bayes method, we use a two-step procedure. First, we fix the regularization parameter and truncation level at optimal values in terms of maximal marginal likelihood. After fixing the regularization parameter and the truncation level, we sample from the posterior distribution of the parameter of interest. A direct implementation of the optimization step for the classification problem is not feasible, so we approximate the posterior distribution using *Laplace approximation*. We provide examples to show the strengths and weaknesses of the empirical Bayes method.

CHAPTER 5
In Chapter 5, we use *variational inference* to approximate the posterior distribution. Variational inference is a method from machine learning and, similar to the empirical Bayes method, it has separate steps for the selection of the hyperparameters and sampling from the approximated posterior distribution. The approximation of the posterior distribution is done through optimization within a class of simpler functions. A major benefit of variational inference is that it allows for *stochastic optimization* to make it applicable to large data. We show this in an example.

1.1. GRAPH THEORY

In this section, we introduce some relevant concepts from graph theory that are used throughout the rest of this dissertation. For a more comprehensive introduction to graph theory see Diestel (1997) and West (1996). For more on graphs within a statistical context see Kolaczyk (2009).

A *graph* $G = (V, E)$ is a structure consisting of a *vertex set* V and an *edge set* E. The vertices $u \in V$ are also sometimes referred to as *nodes* and we usually denote them by $1, \dots, n$, where $\#V = n$. The edges are pairs $\{u, v\}$, where $u, v \in V$. A *subgraph* G' is a graph whose vertex set and edge set are subsets of a given graph G. Given a subset $V' \subset V$ the *vertex-induced subgraph* is $G' = (V', E')$, where $E' \subset E$ are all edges between nodes in V'. If the edges of a graph are ordered pairs, the graph is called a *directed* graph, otherwise the graph is *undirected*. In this dissertation we only consider undirected graphs. In most applications this is reflected by a symmetric relationship between the nodes of the graph, for example having worked together in a social network or protein interaction. If $\{u, v\} \in E$, then we call u and v *adjacent* or *neighbors*. An edge of the form $\{u, u\}$ is called a *loop*. One could allow for multiple edges between nodes, but in this dissertation we only consider graphs without loops or multiple edges, these are also called *simple* graphs. Our method allows for a generalization to graphs with multiple edges or loops or even weighted edges. For simplicity, we do not consider these generalizations in our dissertation. Moreover, we assume that the graph is *connected*. That means there is a path from any vertex u to any other vertex v in the graph, where a path is a sequence of vertices where each vertex is adjacent to the next one. While our algorithms generalize to *disconnected* graphs, we prefer to estimate a function for each *connected component* separately as it is a way to parallelize the computations. The connected components of a graph G are the largest vertex-induced subgraphs that are connected.

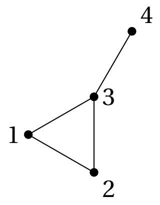


Figure 1.1: A connected simple graph $G = (V,E)$. The vertex set is $V = \{1,2,3,4\}$ and the edges are $E = \{\{1,2\},\{1,3\},\{2,3\},\{3,4\}\}$. The graph G is undirected.

The *adjacency matrix* A is a matrix has entries 1 or 0 in position (u,v) according to whether $\{u,v\} \in E$ or not. It immediately follows that an undirected graph has a symmetric adjacency matrix and if the graph has no loops, the diagonal of the adjacency matrix only has zeros. The *degree* of a vertex v is the number of edges that contain v, it can be recovered from the adjacency matrix as the sum of the column or row corresponding to the vertex v. Note that for directed graphs these correspond to *indegree* and *outdegree* respectively, but for undirected graphs these two sums are equal. For the graph in Figure 1.1, the adjacency matrix is given by

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The vertex degrees are 2, 2, 3, 1. The key concept we will use throughout this dissertation to describe the geometry of a connected simple graph is the *Laplacian matrix* defined as

$$L = D - A,$$

where D is a diagonal matrix containing the vertex degrees and A is the adjacency matrix. It follows that the vector $v = (1,\ldots,1)$ is an eigenvector of L with eigenvalue 0. Moreover, for any vector $x \in \mathbb{R}^n$, we have

$$x^T L x = \sum_{\{i,j\}\in E} (x_i - x_j)^2. \tag{1.1}$$

We can see that the right-hand side in Equation (1.1) is non-negative, so L is positive semi-definite. For the graph in Figure 1.1, the Laplacian matrix is

$$\begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}. \tag{1.2}$$

We are interested in estimating a function $f : V \to \mathbb{R}$ on the vertices of the graph. As the graph has n vertices, a function on the vertices can be interpreted as an element of $\mathbb{R}^n$. Equation (1.1) shows that the Laplacian matrix measures how much a function differs between adjacent vertices. The Laplacian applied to a function on a graph has values

$$(Lf)_i = \sum_{j:\{i,j\}\in E} (f_i - f_j).$$

For example, if a vertex denoted by a triple (x, y, z) has a neighborhood consisting of vertices $(x \pm 1, y \pm 1, z \pm 1)$, then the Laplacian matrix applied to a function on that graph has value

$$\begin{aligned}(Lf)(x,y,z) = 6f(x,y,z) - f(x+1,y,z) - f(x-1,y,z) - f(x,y+1,z) - f(x,y-1,z)\\ - f(x,y,z+1) - f(x,y,z-1),\end{aligned}$$

corresponding to the finite difference for the Laplacian operator in multivariate calculus with unit spacing. This suggests that the Laplacian matrix is a discrete analog of the Laplacian operator, see also Mohar (1991). We can interpret Equation (1.1) as a squared smoothness norm. We use it later on in this dissertation to balance fitting the observed data and the smoothness of the function estimate. In a statistical context, the Laplacian matrix is often used to take into account the geometry of the graph (e.g. Belkin et al. (2004); Kirichenko and Van Zanten (2017); Kolaczyk (2009)).

The geometry of the graph enters through the eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ of the Laplacian matrix. In most examples these will be computed numerically, but for some graphs these can also be derived. For the graph G in Figure 1.1, the eigenvalues of the Laplacian matrix (1.2) are $\lambda_1 = 0$, $\lambda_2 = 1$, $\lambda_3 = 3$ and $\lambda_4 = 4$. The corresponding eigenvectors are $u^{(1)} = (1,1,1,1)$, $u^{(2)} = (-1,-1,0,2)$, $u^{(3)} = (-1,1,0,0)$ and $u^{(4)} = (1,1,-3,1)$.

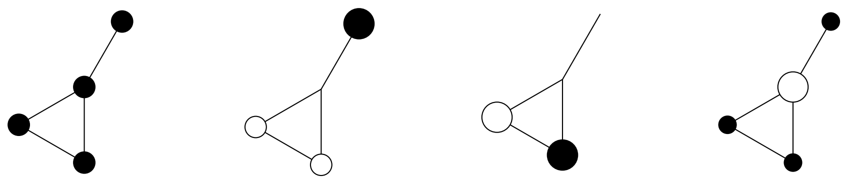


Figure 1.2: The eigenvectors of the graph G in Figure 1.1 visualized as a function on the graph, where the nodes are scaled according to function value and the black or white fill indicates a positive or negative value.

A first example that is used throughout this dissertation is the *path graph*, a connected, simple graph with n nodes, $n - 1$ edges, where two nodes have degree 1 and the other nodes have degree 2.

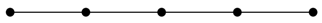


Figure 1.3: Path graph with $n = 5$ nodes.

The eigenvalues of the Laplacian matrix of the path graph are

$$\lambda_j = 4\sin^2\left(\frac{(j-1)\pi}{2n}\right),$$

for $j = 1, \dots, n$. The corresponding eigenvectors are given by

$$u_i^{(j)} = \cos\left(\frac{(i-\frac{1}{2})(j-1)\pi}{n}\right), \tag{1.3}$$

for $i = 1, \dots, n$. These can be verified by direct computation using trigonometric addition and double angle formulas. For more background information about the spectrum of the Laplacian matrix, see Cvetkovic et al. (2009). We will work with unit eigenvectors. The norm of $u^{(j)}$ in Equation (1.3) is $\sqrt{n/2}$ for $j > 1$ and $\sqrt{n}$ for $j = 1$.

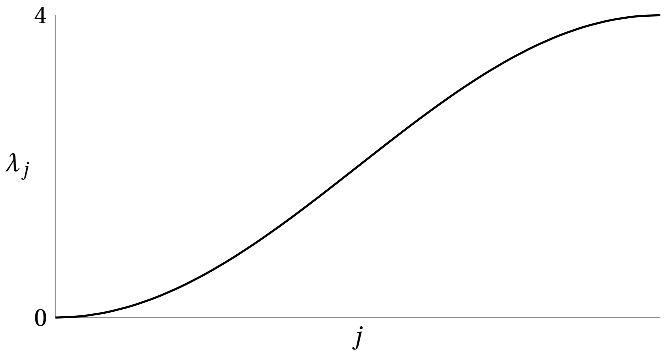


Figure 1.4: The eigenvalues of the path graph plot along the path graph for large n.

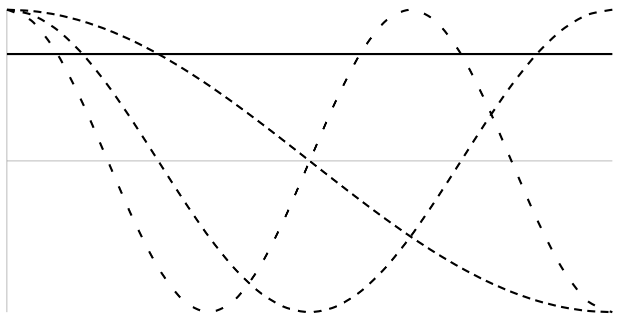


Figure 1.5: The first five eigenvectors of the path graph plot along the path graph for large n. The solid line is the constant eigenvector $u^{(1)} = (1, \dots, 1)$ corresponding to eigenvalue $\lambda_1 = 0$. The densely dashed line is the eigenvector corresponding to λ_2. Increased frequencies, represented by looser dashes, correspond to larger eigenvalues.

In later examples, we will also use the *grid graph*, which is the *graph Cartesian product* of path graphs.

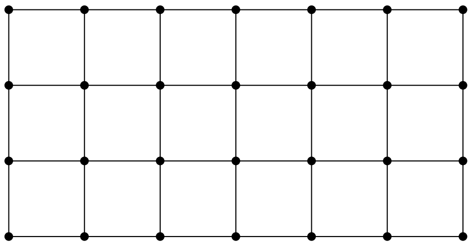


Figure 1.6: A 7×4 grid graph. It is the Cartesian product of a path graph with 7 nodes and a path graph with 4 nodes.

The graph Cartesian product $G_1 \square G_2$ of two graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a graph with vertex set $V_1 \times V_2$ and $u = (u_1, u_2)$ and $v = (v_1, v_2)$ adjacent whenever ($u_1 = v_1$ and $\{u_2, v_2\} \in E_2$) or ($u_2 = v_2$ and $\{u_1, v_1\} \in E_1$).

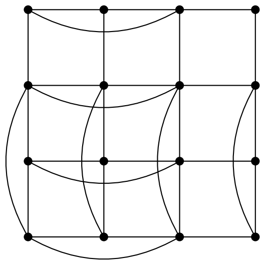


Figure 1.7: The graph Cartesian product $G \square G$, where G is the graph from Figure 1.1.

The eigenvalues of the Laplacian matrix of a graph Cartesian product can be given in terms of eigenvalues of the Laplacian matrices L_1 and L_2 of its components (cf. Theorem 3.5 in Mohar (1991)) as

$$\lambda_i + \mu_j,$$

where $\lambda_i, i = 1, \dots, n_1$ are the eigenvalues of L_1 and $\mu_j, j = 1, \dots, n_2$ are the eigenvalues of L_2. The corresponding eigenvectors are given as a Kronecker product $x^{(i)} \otimes y^{(j)}$, where $x^{(i)}$ is the eigenvector of L_1 with eigenvalue λ_i and $y^{(j)}$ is the eigenvector of L_2 corresponding to μ_j. Recall that the Kronecker product $A \otimes B$ for matrices A and B of arbitrary sizes is a block matrix, with blocks consisting of an element $a_{i,j}B$, where $a_{i,j}$ are the elements of A. Note that the Laplacian matrix of $G_1 \square G_2$ is the Kronecker sum

$$L_1 \oplus L_2 = L_1 \otimes I_{n_2} + I_{n_1} \otimes L_2,$$

where I_n is the $n \times n$ identity matrix. For example, we can see that the graph $G \square G$ from

Figure 1.7 has an eigenvalue $5 = 1 + 4$, corresponding to eigenvector

$$\begin{pmatrix} 1 \\ 1 \\ -3 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} -1 \\ -1 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 2 \\ -1 \\ -1 \\ 0 \\ 2 \\ 3 \\ 3 \\ 0 \\ -6 \\ -1 \\ -1 \\ 0 \\ 2 \end{pmatrix}.$$

The geometry of the graph is related to the eigenvalues of the Laplacian matrix. For example, for a disconnected graph G with K connected components, the Laplacian matrix has eigenvalue zero with multiplicity K. The corresponding eigenvectors are the constant vectors on each of the connected components. For a connected graph G, the second smallest eigenvalue $\lambda_2 > 0$ is also called the *algebraic connectivity* of G and is related to several graph invariants, which can be viewed as measures of connectivity, see, for example, Section 6 of Mohar (1991). For example, λ_2 is a lower bound on the *vertex connectivity*, i.e. the minimum number of nodes, whose deletion would make the graph disconnected.

In our context, we relate the geometry of the graph to the growth of the eigenvalues. We often assume that for some parameter $r \geq 1$, there exists $j_0 \in \mathbb{N}$, $\kappa \in (0,1]$ and $C_1, C_2 > 0$, such that for all n large enough,

$$C_1 \left(\frac{j-1}{n} \right)^{2/r} \leq \lambda_j \leq C_2 \left(\frac{j-1}{n} \right)^{2/r}, \quad \forall j \in \{j_0, \dots, \kappa n\}. \tag{1.4}$$

We will refer to Condition (1.4) as the *geometry condition*. This condition can be verified numerically and for some graphs it can be shown to hold theoretically (see, for example, Kirichenko and Van Zanten (2017)). For the path graph we can use the inequalities $2x/\pi \leq \sin x \leq x$ for $x \in [0,1]$ to see that the geometry condition holds for $r = 1$ with $C_1 = 4$, $C_2 = \pi^2$, $\kappa = 1$ and $j_0 = 1$. In Kirichenko and Van Zanten (2017) it is shown that for d-dimensional regular grid graphs, i.e. the Cartesian product of d path graphs with the same number of nodes, the geometry condition holds for $r = d$, which gives an interpretation to the *geometry number* r as the dimension of the graph. However, note that r does not have to be an integer. Implicitly, we interpret the graph as an element of a sequence of graphs of growing size n. In some applications, such as for the path graph, this is a natural way to think of the graph growing in size.

1.2. STATISTICS

In this section, we give the relevant statistical notation and results that are used throughout this dissertation.

In mathematical statistics, a *statistical model* is defined as a collection of probability distributions. The probability distributions are indexed by a parameter, which is to be inferred from the data. If the unknown parameter is of finite dimension, the problem is called *parametric*. An example of parametric inference is the estimation of the parameters of a linear model, i.e. linear regression. If the parameter of interest is of infinite dimension, we speak of *nonparametric* statistics. Examples of such parameters are functions and distributions. In nonparametric regression, the function of interest does not depend on a finite number of parameters, such as in linear regression, but is an element of an infinite dimensional function space. References for nonparametric statistics are Tsybakov (2009) and Wasserman (2006). Although nonparametric models are typically less restrictive than parametric models, some assumptions need to be posed on the parameter space. This is usually reflected in a choice of bandwidth, smoothness or *regularization parameter* that balances the fit to the data with a notion of complexity, such as regularity or smoothness. The choice of this parameter is well known to be a delicate issue and the label prediction in a graph context is no exception, see Belkin et al. (2006).

A *frequentist* approach to inference is to consider the unknown parameter of the statistical model unknown, but fixed. In *Bayesian* statistics, we treat the unknown parameter as a random variable. The uncertainty about the parameter before observing the data is described in the *prior distribution* on the parameter space. The conditional distribution of the parameter given the observed data is called the *posterior distribution*. It describes the uncertainty of the parameter after observing the data. The posterior distribution can be used to make predictions and the spread of the posterior mass can be used to quantify uncertainty in the predictions. References for a good introduction to Bayesian statistics are Gelman et al. (2014) and Robert (2007). The Bayesian approach can be preferred over the frequentist approach for philosophical reasons. However, many modern applications use Bayesian statistics for its ability to incorporate regularization in complex models via the prior distribution. A reference for Bayesian nonparametrics is Ghosal and Van der Vaart (2017).

We denote the *probability density function (pdf)* of X by $p(x)$ and indicate that X has probability density function p by $X \sim p$. The distribution of X conditional on Y is denoted by $X \mid Y$ and its density by $p(x \mid y) = p(x, y)/p(y)$. We assume $p(y) > 0$, such that this definition makes sense. Moreover, in this dissertation we only deal with densities with respect to the Lebesgue measure, the counting measure and their product measure, in which case we might also talk about a *probability mass function*. With a slight abuse of notation, we use the notation $p(\cdot)$ and $p(\cdot \mid \cdot)$ for all (conditional) probability density functions, it should be clear from the arguments and context which one is meant.

Most calculations in this dissertation revolve around *Gaussian* processes and therefore the *multivariate normal distribution*. We say X is Gaussian or X has a multivariate normal distribution and denote this by $X \sim N(\mu, \Sigma)$, where the pdf is given by

$$p(x) = (\det 2\pi\Sigma)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)},$$

where the parameter μ is the mean vector and Σ the (positive definite) covariance matrix of appropriate dimension. When X is partitioned as (X_1, X_2), a common calculation is the pdf of X_1 conditional on X_2 in terms of the parameters $\mu = (\mu_1, \mu_2)$ and

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}.$$

We use the block matrix inverse

$$\Sigma^{-1} = \begin{pmatrix} (\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1} & -(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1} \\ -\Sigma_{22}^{-1}\Sigma_{21}(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1} & \Sigma_{22}^{-1} + \Sigma_{22}^{-1}\Sigma_{21}(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1} \end{pmatrix}$$

to see that

$$X_1 \mid X_2 \sim N(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}). \tag{1.5}$$

A reference for the use of Gaussian processes in machine learning is Rasmussen and Williams (2006).

We employ a *Bayesian* approach to infer the parameter of interest. In our statistical model of the data X we treat the parameter of interest θ as another random variable and express the *likelihood* of the data as a conditional probability density $p(x \mid \theta)$. We choose a *prior* density $p(\theta)$ and use Bayes' rule to compute the *posterior* density

$$p(\theta \mid x) = \frac{p(x \mid \theta)p(\theta)}{\int p(x \mid \theta)p(\theta)d\theta}.$$

To find the posterior distribution, we usually only consider $p(x \mid \theta)p(\theta)$ as a function of θ. We can always compute the normalization constant afterwards using the law of total probability $\int p(\theta \mid x)d\theta = 1$.

A benefit of using a Gaussian family of distributions is the *conjugacy* in the sense that if we use a prior distribution in this family, then the posterior distribution is also in the Gaussian family. More precisely, if $X \mid \theta \sim N(\theta, \Sigma)$ and we use a prior distribution $\theta \sim N(\mu, \Xi)$, then the posterior distribution is $\theta \mid X \sim N((\Xi^{-1} + \Sigma^{-1})^{-1}(\Xi^{-1}\mu + \Sigma^{-1}X), (\Xi^{-1} + \Sigma^{-1})^{-1})$.

Another useful conjugacy is the normal-inverse gamma conjugacy. The *gamma distribution* has density

$$p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},$$

where $\alpha > 0$ is the shape parameter, $\beta > 0$ is the rate parameter and Γ the gamma function defined by

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt.$$

As we usually don't write the normalization constant, we simply use the notation $X \sim \Gamma(\alpha, \beta)$ for X having a gamma distribution. The gamma distribution is conjugate for the inverse variance (*precision*) of a univariate normal distribution. So, if $X \mid \theta \sim N(\mu, 1/\theta)$ and $\theta \sim \Gamma(\alpha, \beta)$ then $\theta \mid X \sim \Gamma(\alpha + \frac{1}{2}, \beta + \frac{1}{2}(X - \mu)^2)$.

The prior distribution on the parameter θ could, in turn, also depend on an unknown hyperparameter ξ. In this case, we have a *hierarchical Bayesian model*. In a full Bayesian approach, we also endow ξ with a prior $p(\xi)$, such that we can use the three distributions $p(x \mid \theta)$, $p(\theta \mid \xi)$ and $p(\xi)$ to determine the posterior

$$p(\xi, \theta \mid x) \propto p(x \mid \theta) p(\theta \mid \xi) p(\xi).$$

Usually, we are only interested in $p(\theta \mid x)$. We can compute this by integrating the joint posterior over ξ. In principle, we could continue the hierarchy by having ξ also depend on unknown hyperparameters. On the other hand, we can also see that introducing hyperparameters is equivalent to a special choice of prior for θ:

$$p(\theta) = \int p(\theta \mid \xi) p(\xi) d\xi.$$

An alternative to a full Bayesian approach is the *empirical Bayes* approach. In this case, we fix the hyperparameter at ξ^*, where ξ^* maximizes the marginal likelihood

$$p(x \mid \xi) = \int p(x \mid \theta) p(\theta \mid \xi) d\theta.$$

Once we have determined ξ^*, we fix it and infer $p(\theta \mid x)$ in the usual way. This method not only avoids the computation of the integral

$$p(\theta \mid x) \propto \int p(x \mid \theta) p(\theta \mid \xi) p(\xi) d\xi,$$

but also the choice of hyperprior $p(\xi)$.

1.3. BAYESIAN COMPUTATION

In our setting, the data is usually partitioned in observed data and missing data $X = (X^{\text{obs}}, X^{\text{miss}})$. Our two main goals are the computation of the posterior distribution of the parameter of interest based on the observed data $\theta \mid X^{\text{obs}}$ and the computation of the *posterior predictive distribution* $X^{\text{miss}} \mid X^{\text{obs}}$. We can use these distributions to compute functionals of the posterior density. For example, if we observe $X^{\text{obs}} = x$ and we wish to compute the expected value of $g(\theta)$ conditional on x, where g is some function, we can express this as

$$\int g(\theta) p(\theta \mid x) d\theta. \tag{1.6}$$

In some special cases, this integral can be computed explicitly, but generally a numerical method has to be used. Specifically in our setting, the posterior distribution is a high-dimensional distribution, so grid-based methods might not be feasible. As an alternative, a *Monte Carlo method* can be used. Such methods use a sample from the posterior distribution $\theta^{(1)}, \ldots, \theta^{(N)} \sim p(\theta \mid x)$ to estimate the expectation (1.6) as

$$\frac{1}{N} \sum_{t=1}^{N} g(\theta^{(t)}).$$

An advantage is that the accuracy of this estimate depends on the sample size N and not on the dimension of the parameter θ.

To sample from the posterior distribution several techniques exist. For an overview of methods to generate random numbers see Devroye (1986). We assume that we have a *random number generator* able to generate an independent sample from several well-known distributions, such as the normal distribution, the gamma distribution and the *standard uniform distribution*, i.e. the distribution with pdf $p(x) = 1$ for $x \in (0,1)$. Standard random number generators for these distributions are usually included in statistical software packages such as R (R Core Team, 2018). To sample from other distributions, we can use exact simulation methods, such as the *inversion method* and the *rejection method*, see for example Madras (2002). To illustrate the rejection method, consider sampling from a normal distribution with mean μ and variance σ^2 conditioned to be positive. To achieve this, we generate a proposal from $N(\mu,\sigma^2)$ and only accept the proposal if it is positive.

In more complex models, direct simulation from the posterior distribution is often not possible. In these cases, *Markov chain Monte Carlo (MCMC)* methods often provide a solution. The basic idea is to generate a sequence $\theta^{(0)},\theta^{(1)},\ldots$ recursively, such that, for large t, $\theta^{(t)}$ approximately has the desired distribution. The recursive generation only uses $\theta^{(t)}$ and new random variables to determine $\theta^{(t+1)}$, but it does not use $\theta^{(s)}$ for $s < t$, i.e. the sequence is a Markov chain. As the Markov chain approximates the target distribution, we might want to discard the first part of the sequence. This is called the *burn-in* period. A consequence of the recursive generation is that $\theta^{(t+1)}$ and $\theta^{(t)}$ are generally dependent. To reduce the dependence in the sample, we can, for example, only keep one in every ten iterations and discard the rest. This is called *thinning*. For more techniques and background information about MCMC methods, see Brooks et al. (2011). For these methods in Bayesian context, see for example Gelman et al. (2014).

The *Gibbs sampler* is one way to recursively generate random variables approximating a target distribution. Suppose we can partition $\theta = (\theta_1,\theta_2)$ and we wish to sample from $p(\theta_1,\theta_2)$. We can achieve this by iteratively sampling from $p(\theta_1\,|\,\theta_2)$ and $p(\theta_2\,|\,\theta_1)$:

Algorithm 1.1 Gibbs sampler.

```
Input: Starting value $\theta^{(0)} = (\theta_1^{(0)},\theta_2^{(0)})$.
Output: Sample $\theta^{(0)},\theta^{(1)},\ldots$ approximately distributed as $p(\theta_1,\theta_2)$.
1: for $t = 1,2,\ldots$ do
2:     Sample $\theta_1^{(t)} \sim p(\theta_1\,|\,\theta_2^{(t-1)})$.
3:     Sample $\theta_2^{(t)} \sim p(\theta_2\,|\,\theta_1^{(t)})$.
4: end for
```

To illustrate the Gibbs sampler, consider $\theta = (\theta_1,\theta_2)$ having a bivariate normal distribution with mean $(0,0)$, marginal variance 1 and correlation ρ. We can use Equation (1.5) to see that $\theta_1\,|\,\theta_2 \sim N(\rho\theta_2, 1-\rho^2)$ and $\theta_2\,|\,\theta_1 \sim N(\rho\theta_1, 1-\rho^2)$. We can use these conditionals to iteratively sample from the joint distribution of θ. An example is given in Figure 1.8. Note that to sample from a multivariate normal distribution $N(\mu,\Sigma)$, we usually use the Cholesky decomposition $\Sigma = CC^T$ and sample $\theta = \mu + CZ$, where Z has a standard normal distribution.

Algorithm 1.1 naturally generalizes to partitions in more than two components.

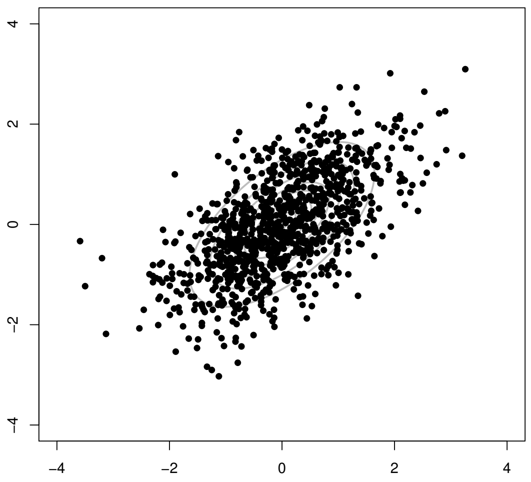


Figure 1.8: Approximate sample from a bivariate normal distribution, generated using a Gibbs sampler. The grey lines in the background are the contour lines for the actual pdf.

Gibbs sampling is particularly applicable in hierarchical Bayesian models. In these models, the data has likelihood $p(x|\theta)$ and the prior distribution $p(\theta|\xi)$ depends on some hyperparameter ξ, which has a hyperprior $p(\xi)$. Both the parameter of interest θ and the hyperparameter ξ can be inferred simultaneously from the data. We are interested in the joint posterior $p(\xi,\theta|x)$. The Gibbs sampler approximates this distribution by sampling iteratively from the conditional pdfs $p(\theta|\xi,x)$ and $p(\xi|\theta)$. This allows for computations in hierarchical models where the joint posterior $p(\xi,\theta|x)$ is untractable, but there exists partial conjugacy for the prior conditional on the hyperparameter.

When partial conjugacy cannot be exploited in a Gibbs sampler, the more general *Metropolis-Hasitngs algorithm* might provide a solution:

Algorithm 1.2 Metropolis-Hastings algorithm.

```
Input: Starting value $\theta^{(0)}$. Proposal density $q(\cdot,\cdot)$. Target density $p(\cdot)$.
Output: Sample $\theta^{(0)}, \theta^{(1)}, \dots$ approximately distributed as $p(\theta)$.
 1: for $t = 1, 2, \dots$ do
 2:     Sample a proposal $\theta' \sim q(\cdot, \theta^{(t-1)})$.
 3:     Calculate
            $r \leftarrow \frac{p(\theta') q(\theta^{(t-1)}, \theta')}{p(\theta^{(t-1)}) q(\theta', \theta^{(t-1)})}.$
 4:     Sample $u \sim U(0,1)$.
 5:     if $u \le r$ then
 6:         Accept proposal, $\theta^{(t)} \leftarrow \theta'$.
 7:     else
 8:         Reject proposal, $\theta^{(t)} \leftarrow \theta^{(t-1)}$.
 9:     end if
10: end for
```

The *proposal density* $q(\theta', \theta)$ does not have to be symmetric and could, in general, also depend on the iteration t. It is the transition density of the proposed move from θ to θ'. Common choices are *independent proposals* and *random walk proposals*. For independent proposals, the proposal density $q(\theta', \theta)$ in state θ, does not depend on θ. For random walk proposals, the proposal has the form $\theta' = \theta + \epsilon$, where ϵ is independent of θ, this implies $q(\theta', \theta)$ has the form $f(\theta' - \theta)$ for some function f. The *acceptance probability* of the proposed move to θ' from state θ is $a(\theta', \theta) = \min\{1, r(\theta, \theta')\}$ in Algorithm 1.2.

To illustrate the Metropolis-Hastings algorithm, we look at the example from Figure 1.8. Now we use a Metropolis-Hasting algorithm to sample from the bivariate normal distribution. We use a random walk proposal $\theta + \epsilon$, where $\epsilon \sim N(0,1)$. The results are in Figure 1.9.

The Metropolis-Hastings update is constructed such that the resulting sample is a Markov chain, and that the target distribution is *reversible* with respect to the Markov chain. A proof and technical details can be found in Tierney (1998). Intuitively, we should design the proposal distribution such that the Markov chain is *irreducible*, i.e. it can reach all states from any starting point, with positive probability. Then, we verify reversibility through the *detailed balance relation* (cf. Brooks et al. (2011); Tierney (1998))

$$p(\theta) q(\theta', \theta) a(\theta', \theta) = p(\theta') q(\theta, \theta') a(\theta, \theta'). \tag{1.7}$$

As a last generalization, we consider *reversible jump Markov chain Monte Carlo*. Suppose the state space of θ is $\mathscr{C} = \bigcup_{k \in \mathscr{K}} \mathscr{C}_k$, where $\mathscr{K}$ is a discrete set of model indices, for example $\mathscr{K} = \mathbb{N}$, and $\mathscr{C}_k = \{k\} \times \mathbb{R}^{n_k}$. This means that given k, θ has dimension n_k and we can denote $\theta = (k, \xi)$. In the context of Bayesian model selection this interpretation is

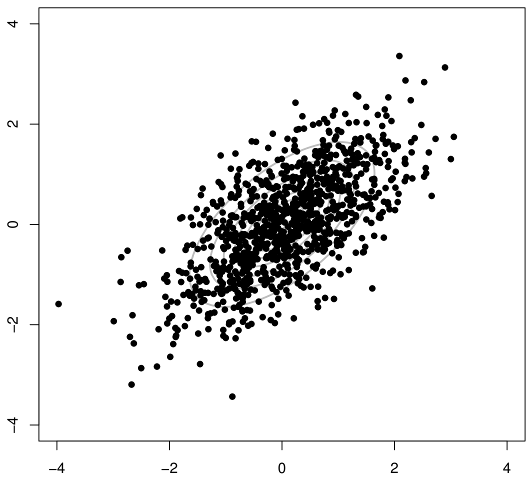


Figure 1.9: Approximate sample from a bivariate normal distribution, generated using a Metropolis-Hasting sampler. The grey lines in the background are the contour lines for the actual pdf.

very useful. The detailed balance relation (1.7) can still be interpreted in the case of jump across dimensions and has to be verified. A precise treatment of reversible jump MCMC is given in Green (1995), in addition to Tierney (1998) and Brooks et al. (2011). A tutorial with minimal measure theoretic notation is Waagepetersen and Sorensen (2001).

In this dissertation, we only use a special case of reversible jump MCMC, where the models are nested and the change of dimension only consists of adding or deleting a component of the parameter vector $\xi \in \mathbb{R}^k$. This is equivalent to considering a large parameter Ξ that includes all possible components of ξ, and to set some components to zero if they are omitted from the model. We can achieve this using a Metropolis-Hastings algorithm, where we allow the proposal distributions to have a point mass at zero. This will result in an equivalent algorithm to the reversible jump approach.

Algorithm 1.3 Reversible jump MCMC.

```
Input: Starting value $\theta^{(0)} = (k^{(0)}, \xi^{(0)})$. Proposal density $q(\cdot,\cdot)$, prior density $p(k)$, likelihood $p(\xi \mid k)$. Target density $p(k,\xi)$.
Output: Sample $\theta^{(0)}, \theta^{(1)}, \dots$ approximately distributed as $p(\theta)$.
1: for $t = 1,2,\dots$ do
2:     Sample a proposal $k' \sim q(\cdot, k^{(t-1)})$.
3:     Sample $\xi \sim p(\cdot \mid k')$.
4:     Calculate
           $r \leftarrow \frac{p(k')q(k^{(t-1)},k')}{p(k^{(t-1)})q(k',k^{(t-1)})}$.
5:     Sample $u \sim U(0,1)$.
6:     if $u \le r$ then
7:         Accept proposal, $\theta^{(t)} \leftarrow (k', \xi)$.
8:     else
9:         Reject proposal, $\theta^{(t)} \leftarrow \theta^{(t-1)}$.
10:    end if
11: end for
```

To illustrate the use of reversible jump MCMC in a Bayesian context, we consider the following polynomial regression example. We generate points (x_i, y_i) for $i = 1, \dots, n$, where $x_i \sim U(0,1)$ and $y_i = f(x_i) + \epsilon_i$, where $\epsilon_i \sim N(0,1)$ are independent and

$$f(x) = \frac{\sin(12(x+0.2))}{x+0.2}. \tag{1.8}$$

Our goal to estimate the unknown function $f(x)$ using an orthonormal basis of polynomials $u^{(j)}$ for $j = 0,1,2,\dots$, where polynomial $u^{(j)}$ has degree j. Let U be the matrix with columns $u^{(j)}$ evaluated at $x_1, \dots, x_n$, and U_k its restriction to the columns up to degree k. We estimate f with

$$U_k \xi = \sum_{j=0}^{k} \xi_j u^{(j)},$$

where the degree k and the coefficients ξ are unknown. We use a hierarchical Bayesian model using the following prior: given the degree k, the coefficients ξ have a Gaussian distribution with mean 0 and variance σ^2. The polynomial degree k has a uniform distribution on $\{0,1,\dots,15\}$. We consider a maximal degree, because we do not want to overfit the data. Our hierarchical Bayesian model can be summarized as

$$\begin{aligned} y \mid \xi, k &\sim N(U_k \xi, I_n), \\ \xi \mid k &\sim N(0, \sigma^2 I_{k+1}), \\ k &\sim \text{Uniform}\{0, \dots, 15\}, \end{aligned} \tag{1.9}$$

where I_n and I_k are identity matrices of dimensions $n \times n$ and $(k+1) \times (k+1)$. Given the observations $D = \{(x_i, y_i) : i = 1, \dots, n\}$, we use Algorithm 1.3 to sample from $p(k, \xi \mid D)$. If we're in state k, we use a random walk proposal $q(\cdot, k)$ assigning probabilities 0.25, 0.5, 0.25 to $k-1$, k, $k+1$. Note that the target density is $p(k, \xi \mid D)$, so we also have to consider $p(\xi \mid k, D)$ and $p(k \mid D)$ instead of $p(\xi \mid k)$ and $p(k)$. Using the model (1.9) and properties of the normal distribution, we can find

$$\xi \mid k, D \sim N\left(\frac{\sigma^2}{1+\sigma^2} U_k^T y, \frac{\sigma^2}{1+\sigma^2}\right)$$

and

$$p(k \mid D) \propto (1+\sigma^2)^{-\frac{k}{2}} e^{\frac{1}{2} \frac{\sigma^2}{1+\sigma^2} y^T U_k U_k^T y}.$$

Figure 1.10 shows a sample from the posterior $k, \xi \mid D$ generated by Algorithm 1.3. The left plot shows a sample of 20 draws from the posterior of $f \mid D$. In the middle plot we have summarized a sample of size 1000 to construct point-wise credible intervals, depicted by the grey area. The right plot shows a histogram of the posterior degree of the polynomial k. We observe that it concentrates around the posterior mean 6.9.

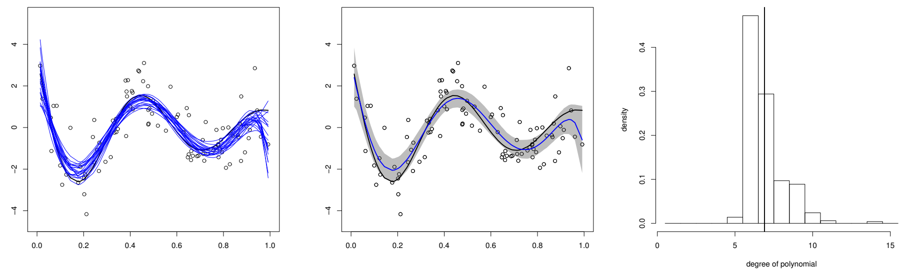


Figure 1.10: An illustration of Algorithm 1.3 in the context of polynomial regression. The data D consists of $n = 100$ points. They are used to estimate the true underlying function in Equation (1.8) using polynomials with random degree k. A sample of size 1000 is generated using Algorithm 1.3 to approximate the posterior distribution. The actual function is the black line. Left: A sample of size 20 from the posterior $f \mid D$. Middle: Point-wise credible intervals (gray area) and the posterior mean (blue line). Right: Histogram of the posterior of the degree of the polynomial and the posterior mean (black line).

Gibbs, Metropolis-Hastings and reversible jump updates can be used as building blocks to construct MCMC methods to sample from the posterior distribution.

1.4. PREDICTION PROBLEM

We have problems in mind in which the graph is given by the application context. Concretely, we consider a connected, simple graph $G = (V, E)$, with $\#V = n$ vertices, which we denote by $1,2,\ldots,n$. Associated to every vertex i is a noisy label y_i, which, in case of regression, can be a real number or, in case of classification, a discrete value. The available data are noisy observations of some of the labels. So, we observe only a subset $Y^{\text{obs}} \subset \{y_1,\ldots,y_n\}$, generated in an arbitrary way, but independent of the value of the labels. Specifically, we assume that for some arbitrary distribution μ on the collection 2^V of subsets of vertices, a set of vertices $I^{\text{obs}} \subset V$ is drawn and that we see which vertices were selected and what the corresponding noisy labels are. In other words, the observed data is $D = \{(i, y_i) : i \in I^{\text{obs}}\}$. The idea is that typically, the location of a given vertex in the graph, in combination with (noisy) information about the labels of vertices close to it, should have predictive power for the label of the vertex of interest. Hence, successful prediction of labels should be possible to some degree.

In the regression problem, the labels are independent normal random variables, with

$$y_i \sim N(f_i, \sigma^2),$$

where

$$f : V \to \mathbb{R}$$

is an unobserved function on the vertices of the graph. We can consider σ^2 given and fixed or as an unknown parameter.

In the binary classification problem, the labels are independent Bernoulli random variables, with

$$P(y_i = 1) = 1 - P(y_i = 0) = \ell(i),$$

where

$$\ell : V \to (0,1)$$

is an unobserved function on the vertices of the graph. It is called the *soft label function*. We write $\ell = \Phi(f)$ for some function $f : V \to \mathbb{R}$ and Φ the *probit link function*, i.e. the cdf of a standard normal distribution. The underlying idea is that the real hard labels $h(i)$ of the vertices are obtained by thresholding the soft labels at level $1/2$, i.e. $h(i) = 1_{\ell(i)>1/2}$. The y_i are noisy versions of the real hard labels $h(i)$, in the sense that they can be wrong with some positive probability. Specifically, it holds in this setup that $P(y_i \neq h(i)) = |h(i) - \ell(i)|$. An equivalent formulation of the classification problem, using an additional layer of *latent variables* $z = (z_1,\ldots,z_n)$ is given by

$$y_i = 1_{z_i>0}$$

and

$$z \sim N(f, I),$$

where I is the $n \times n$ identity matrix, cf. Albert and Chib (1993); Choudhuri et al. (2007). Compared to the regression problem, this function f plays a similar role as parameter of interest, but in the classification problem and additional layer z is between the function f and the observation y.

In both setups, we take a nonparametric Bayesian approach. It is nonparametric in the sense that we do not assume that f belongs to some low-dimensional, for instance (generalized) linear family of functions. We put a prior on f, which takes the geometry of the graph into account via the Laplacian matrix and a *regularization parameter*.

This hyperparameter controls the bias-variance trade-off, in the sense that it balances fit to the data with a sense of smoothness of the function f. Similar methods employing *Laplacian regularization* are Belkin et al. (2004), Kolaczyk (2009) and Kirichenko and Van Zanten (2017).

In the following chapters, we explore different methods to compute the posterior distribution $f \mid D$. This posterior distribution can be used to predict the unobserved labels. We focus mainly on the classification problem as this is a more difficult problem and has more applications. However, we use the regression problem as a simple example to illustrate our proposed methods. In Chapter 2, we employ a full hierarchical Bayesian approach, resulting in an MCMC algorithm to sample from the joint posterior distribution of the function f and the regularization parameter. In Chapter 3, we change our prior to include a truncation level for more flexibility and computational efficiency and still consider a full hierarchical Bayesian approach. The result is a faster MCMC algorithm to sample from the joint posterior distribution of the function f, the truncation level and the regularization parameter. In Chapter 4, we use *empirical Bayes* methods to select the truncation level and the regularization parameter. These hyperparameters are then fixed and this greatly simplifies sampling from the posterior distribution of f. However, to select the hyperparameters, an optimization problem has to be solved. In Chapter 5, we use *variational inference* to approximate the posterior distribution. We select the hyperparameters in a first step and sample from the approximated posterior distribution of f in a second step. The use of *stochastic optimization* makes this method scalable to large data.

2

HIERARCHICAL BAYES

This chapter describes an implementation of a nonparametric Bayesian approach to solve regression and binary classification problems on graphs. We consider a hierarchical Bayesian approach with a randomly scaled Gaussian prior. The prior uses the Laplacian matrix of the graph to take the underlying geometry of the graph into account. A method based on a theoretically optimal prior and a more flexible variant using partial conjugacy are proposed. Two simulated data examples and two examples using real data are used in order to illustrate the proposed methods.

In this chapter, we consider prediction problems that can be seen as regression or classification problems on graphs. These arise in several applied settings, for instance in the prediction of the biological function of a protein in a protein-protein interaction graph (e.g. Kolaczyk (2009); Nariai et al. (2007); Sharan et al. (2007)), or in graph-based semi-supervised learning (e.g. Belkin et al. (2004); Sindhwani et al. (2007)). We have problems in mind in which the graph is given by the application context and the graph has vertices of different types, coded by vertex labels that can have two possible values, say for the classification problem. For the regression problem the vertices have a real-valued label. The available data are noisy observations of some of the labels. The goal of the statistical procedure is to predict the vertices correctly, including those for which there is no observation available. The idea is that typically, the location of a given vertex in the graph, in combination with (noisy) information about the labels of vertices close to it, should have predictive power for the label of the vertex of interest. Hence, successful prediction of labels should be possible to some degree.

Several approaches for graph-based prediction have been considered in the literature. In this chapter, we investigate a nonparametric full Bayesian procedure that was recently considered in Kirichenko and Van Zanten (2017) and that has so far only been studied theoretically. The Bayesian approach proposed in Kirichenko and Van Zanten

Parts of this chapter have been published in Computational Statistics & Data Analysis **120**, 111-131 (2018) (Hartog and Van Zanten, 2018).

(2017) consists of endowing the label function f with a prior distribution and determining the corresponding posterior. The priors we consider are described in detail in the next section.

The posterior distribution for f that results from a Bayesian analysis can be used for instance for prediction. For the priors we consider, the computation of the posterior mode is closely related to the computation of a kernel-based regression estimate with a kernel based on the Laplacian matrix associated with the graph (e.g. Ando and Zhang (2007); Belkin et al. (2004); Kolaczyk (2009); Smola and Kondor (2003); Zhu and Hastie (2005)). In that sense the method we consider is close to those used in the cited papers. A benefit of the full Bayesian framework is that the spread of the posterior may be used to produce a quantification of the uncertainty in the predictions. Moreover, we specifically consider a full hierarchical Bayesian procedure, because of its capability, when properly designed at least, to automatically let the data determine the appropriate value of crucial tuning parameters.

As is well known, the choice of bandwidths, smoothness, or regularization parameters in nonparametric methods is a delicate issue in general. The graph context is no exception in this regard and it is recognized that it would be beneficial to have a better understanding of how to choose the regularization parameters (e.g. Belkin et al. (2006)). The theoretical results in Kirichenko and Van Zanten (2017) indicate how the performance of nonparametric Bayesian prediction on graphs depends on both the geometry of the underlying graph and on the smoothness of the (unobserved) function f. Moreover, for a certain family of Gaussian process priors Kirichenko and Van Zanten (2017) gives guidelines for choosing the hyperparameters in such a way that asymptotically optimal performance is obtained.

The aim of this chapter is to provide an implementation of nonparametric Bayesian prediction on graphs using Gaussian priors based on the Laplacian matrix of the graph. Moreover, we investigate numerically the influence of the geometry of the graph and the smoothness of the function of interest, motivated by the asymptotics given in Kirichenko and Van Zanten (2017). In this manner we arrive at recommended choices for tuning parameters and hyperpriors that are in line with the theoretical guarantees and that are also computationally convenient.

The rest of this chapter is organized as follows. In the next section, a description of the priors we consider for the classification problem are given. Algorithms to sample from the posterior distribution are given in Section 2.2 and computational aspects are discussed in Section 2.3. In Section 2.4 we present numerical experiments. We first apply and test the procedure on two simulated data sets, one involving the path graph and one a small-world graph, respectively. Next, we study the performance on real data, considering the problems of predicting the function of a protein in a protein-protein interaction graph, and classifying hand-written digits using a nearest neighbor graph. Whereas the rest of this chapter focuses on the classification problem, in Section 2.5 we present our results in the context of the regression problem. Concluding remarks are given in Section 2.6.

2.1. OBSERVATION MODEL AND PRIORS

2.1.1. OBSERVATION MODEL

We start with a connected, simple undirected graph $G = (V, E)$, with $\#V = n$ vertices denoted by $V = \{1, 2, \ldots, n\}$. Associated to every vertex i is a noisy hard label y_i. We assume the y_i's are independent Bernoulli variables, with

$$P(y_i = 1) = 1 - P(y_i = 0) = \ell(i),$$

where $\ell : V \to (0,1)$ is an unobserved function on the vertices of the graph, the so-called soft label function. We observe only a subset $Y^{\text{obs}} \subset \{y_1, \ldots, y_n\}$ of all the noisy labels. This can be a random subset of all the $\{y_1, \ldots, y_n\}$, generated in an arbitrary way, but independent of the values of the labels. Note that in this setup we either observe the label of a vertex or not, so multiple observations of the same vertex are not possible.

2.1.2. PRIOR ON THE SOFT LABEL FUNCTION

Our prediction method consists in first inferring the soft label function ℓ from Y^{obs} and subsequently predicting the hard labels by thresholding. We take a Bayesian approach that is nonparametric, in the sense that we do not assume that ℓ belongs to some low-dimensional, for instance generalized linear family of functions.

PRIOR ON ℓ

To put a prior on ℓ we first use the probit link Φ (i.e. the cdf of the standard normal distribution) to write $\ell = \Phi(f)$ for some function $f : V \to \mathbb{R}$ and then put a prior on f. To achieve a form of Laplacian regularization, which takes the geometry of the graph into account (e.g. Belkin et al. (2004); Kirichenko and Van Zanten (2017); Kolaczyk (2009)) we employ a Gaussian prior with a covariance structure that is based on the Laplacian L of the graph G. Recall that this is the matrix defined as $L = D - A$, where D is the diagonal matrix of vertex degrees and A is the adjacency matrix of the graph. (See for instance Cvetkovic et al. (2009) for background information.)

We want to consider a Gaussian prior on f with a fixed power of the Laplacian matrix as precision (inverse covariance) matrix. As the Laplacian matrix has eigenvalue 0 however, it is not invertible. Therefore we add a small number $1/n^2$ to all eigenvalues of L to overcome this problem. By Theorem 4.2 of Mohar (1991), we know that the smallest positive eigenvalue λ_2 of L satisfies $\lambda_2 \geq 4/n^2$, which motivates this choice. To make the prior flexible enough we add a multiplicative scale parameter $c > 0$ as well. Together, this results in a Gaussian prior for f with zero mean and precision matrix $c(L + n^{-2}I)^q$, where $q, c > 0$. We then have

$$\begin{aligned} \ell &= \Phi(f), \\ f|c &\sim N(0, (c(L + n^{-2}I)^q)^{-1}). \end{aligned}$$

To make the connection with kernel-based learning, we note that in the corresponding regularized kernel-based regression model, the matrix $(L + n^{-2}I)^q$ corresponds to the kernel and the scale parameter c to the regularization parameter controlling the trade-off between fitting the observed data and the smoothness of the function estimate, as measured by the squared smoothness norm $f^T(L + n^{-2}I)^q f$.

PRIOR ON c

As with all nonparametric methods, the performance of our procedure will depend crucially on the choice of the hyperparameters c and q, which control the bias-variance trade-off. The correct choices of these parameters depends in principle on properties of the unobserved function f (or equivalently, the function ℓ). Theoretical considerations in Kirichenko and Van Zanten (2017) have shown that good performance can be obtained across a range of regularities of f by fixing q at an appropriate level, and putting a prior on the hyperparameter c, so that we obtain a hierarchical Bayesian procedure. The choices for the prior on c and for q that were shown to work well in Kirichenko and Van Zanten (2017) depend on the geometry of the graph G. A main goal of the present chapter is to investigate numerically whether this dependence is indeed visible when the method is implemented and to investigate choices for q and c that yield good performance and are computationally convenient as well.

The geometry of the graph G enters through the eigenvalues of the Laplacian, which we denote by $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_n$. (See e.g. Chapter 7 of Cvetkovic et al. (2009) for the main properties of the spectrum of L.) In Kirichenko and Van Zanten (2017) the theoretical performance of nonparametric Bayesian methods on graphs is studied under the assumption that for some parameter $r \ge 1$, there exist $j_0 \in \mathbb{N}$, $\kappa \in (0,1]$ and $C_1, C_2 > 0$ such that for all n large enough,

$$C_1\left(\frac{j-1}{n}\right)^{2/r} \le \lambda_j \le C_2\left(\frac{j-1}{n}\right)^{2/r}, \qquad \forall j \in \{j_0,\dots,\kappa n\}. \tag{2.1}$$

This condition can be verified numerically for a given graph (as is done for instance for certain protein-protein interaction and small world graphs in Kirichenko and Van Zanten (2017)) and can be shown to hold theoretically for instance for graphs that look like regular grids or tori of arbitrary dimensions.

In our notation, the hyperprior for the regularization parameter c that was shown to have good theoretical properties in Kirichenko and Van Zanten (2017) (under the geometry condition (2.1)) is the prior with density p given by

$$p(c) \propto c^{-r/(2q)-1} e^{-nc^{-r/(2q)}}, \qquad c > 0. \tag{2.2}$$

If the true (unknown) soft label function ℓ has regularity $\beta > 0$, defined in an appropriate, Sobolev-type sense, then this choice for c guarantees that the posterior contracts around the truth at the optimal rate, provided the hyperparameter q has been chosen such that $q \ge \beta$. Below we investigate the effect of choosing q or r too high or too low relative to these optimal choices.

To better understand how crucial it is to use the prior (2.2) and to set its hyperparameters just right, we compare it to a slightly simpler choice that is natural here, which is a gamma prior on c with density

$$p(c) \propto c^{a-1} e^{-bc}, \qquad c > 0, \tag{2.3}$$

for certain $a, b > 0$. This choice is computationally convenient due to the usual normal-inverse gamma partial conjugacy (see e.g. Choudhuri et al. (2007); Liang et al. (2007) in the context of our setting). It introduces two more hyperparameters a and b. The authors of Choudhuri et al. (2007) and Liang et al. (2007) mention $a = b = 0$, corresponding to Jeffreys prior, which is improper, but does not result in any computational restrictions. We will see in the numerical experiments that this choice is a reasonable one in our setting as well.

To distinguish between the two priors (2.2) and (2.3) in the paper we always call (2.3) the *ordinary gamma* prior for c, and (2.2) the *generalized gamma* prior for c.

We remark that since we are considering two competing priors on c, we could in principle consider some form of (Bayesian) model selection. However, we will see in the numerical experiments that the ordinary gamma prior generally performs better than the generalized gamma. Since the ordinary gamma prior is also preferable from the computational perspective, we would recommend to use the ordinary gamma instead of a combined method.

2.1.3. LATENT VARIABLES AND MISSING LABELS

Combining what we have so far, we obtain a hierarchical model that can be described as follows:

$$\begin{aligned} y_i \mid f, c &\sim \text{independent Bernoulli}(\Phi(f_i)), \qquad i = 1, \dots, n, \\ f \mid c &\sim N(0, (c(L + n^{-2} I)^q)^{-1}), \\ c &\sim p \quad \text{given by (2.2) or (2.3).} \end{aligned}$$

As is well known (cf. Albert and Chib (1993)) an equivalent formulation using an additional layer of latent variables $z = (z_1, \dots, z_n)$ is given by

$$\begin{aligned} y_i &= 1_{z_i > 0}, \qquad i = 1, \dots, n, \\ z \mid f, c &\sim N(f, I), \\ f \mid c &\sim N(0, (c(L + n^{-2} I)^q)^{-1}), \\ c &\sim p \quad \text{given by (2.2) or (2.3).} \end{aligned}$$

This is a more convenient representation which we will use in our computations.

We consider the situation in which we do not observe all the labels y_i, but only a certain subset Y^{obs}. The labels that we observe, are only observed once. The precise mechanism that determines which y_i's we observe and which ones are missing is not important for the algorithm we propose. We only assume that it is independent of the other elements of the model. Specifically, we assume that for some arbitrary distribution μ on the collection 2^V of subsets vertices, a set of vertices $I^{\text{obs}} \subset V$ is drawn and that we see which vertices were selected and what the corresponding noisy labels are. In other words, the observed data is $D = \{(i, y_i) : i \in I^{\text{obs}}\}$.

All in all, the full hierarchical scheme we will work with is the following:

$$\begin{aligned} D &= \{(i, y_i) : i \in I^{\text{obs}}\}, \\ I^{\text{obs}} &\sim \mu, \\ y_i &= 1_{z_i>0}, \qquad i = 1, \ldots, n, \\ z \,|\, f, c &\sim N(f, I), \\ f \,|\, c &\sim N(0, (c(L + n^{-2} I)^q)^{-1}), \\ c &\sim p \quad \text{given by (2.2) or (2.3).} \end{aligned} \tag{2.4}$$

Our goal is to compute the posterior $f \,|\, D$ and to use it to predict the unobserved labels.

2.2. SAMPLING SCHEME

We use the latent variable approach as in Albert and Chib (1993); Choudhuri et al. (2007), for instance, and implement a Gibbs sampler, as in Algorithm 1.1, to sample from the posterior $f \,|\, D$ in the setup (2.4). This involves sampling repeatedly from the conditionals $p(z \,|\, c, f, D)$, $p(f \,|\, c, z, D)$ and $p(c \,|\, z, f, D)$. We detail these three steps in the following subsections.

2.2.1. SAMPLING FROM $p(z \,|\, c, f, D)$

By construction, we have that given D, the latent variables z_i, $i \not\in I^{\text{obs}}$ corresponding to the missing observations are independent of those corresponding to the observed labels. We simply have that given c, f and D, the variables z_i, $i \not\in I^{\text{obs}}$, are independent $N(f_i, 1)$-variables.

As for the latent variables corresponding to the observed labels, we have, by (2.4), that the z_i's are independent given c, f and D and

$$p(z_i \,|\, c, f, D) = p(z_i \,|\, c, f, y_i) \propto p(y_i \,|\, z_i) p(z_i \,|\, f, c) \propto 1_{z_i \text{ matches } y_i} e^{-\frac{1}{2}(z_i - f_i)^2},$$

where we say that z_i matches y_i if $y_i = 1_{z_i>0}$. For $y_i = 1$, this describes the $N(f_i, 1)$-distribution, conditioned to be positive. We denote this distribution by $N_+(f_i, 1)$. For $y_i = 0$ it corresponds to the $N(f_i, 1)$-distribution, conditioned to be negative. We denote this distribution by $N_-(f_i, 1)$.

Put together, we see that given c, f, and D, the z_i are independent and

$$z_i \,|\, c, f, D \sim \begin{cases} N(f_i, 1), & \text{if } i \not\in I^{\text{obs}}, \\ N_+(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 1, \\ N_-(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 0. \end{cases}$$

We note that generating normal random variables conditioned to be positive or negative can for example be done by a simple rejection algorithm or inversion (e.g. Devroye (1986)).

2.2.2. SAMPLING FROM $p(f \mid c, z, D)$

Since given z, we know all the y_i's, and I^{obs} is independent of all other elements of the model, we have $p(f \mid c,z,D) = p(f \mid c,z)$. Next, we have

$$p(f \mid c,z) \propto p(z \mid f,c)\, p(f \mid c).$$

By plugging in what we know from (2.4) we get

$$p(f \mid c,z) \propto e^{-\frac{1}{2} f^T (I + c(L+n^{-2}I)^q) f + z^T f}.$$

Completing the square, it follows that

$$f \mid c,z \sim N\big((I + c(L+n^{-2}I)^q)^{-1} z, (I + c(L+n^{-2}I)^q)^{-1}\big). \tag{2.5}$$

2.2.3. SAMPLING FROM $p(c \mid z, f, D)$

Again, we use that given z we know all the y_i's, and that I^{obs} is independent of everything else, which gives $p(c \mid z,f,D) = p(c \mid z,f)$. Since given f, z is independent of c, we have

$$p(c \mid z,f) \propto p(f \mid c)\, p(c). \tag{2.6}$$

Now, the method for (approximate) sampling from $c \mid z,f$ depends on the choice of the prior for c.

ORDINARY GAMMA PRIOR FOR c

If the prior density for c is the ordinary gamma density given by (2.3) we have the usual normal-inverse gamma conjugacy. Indeed, then we have

$$p(f \mid c)\, p(c) \propto c^{n/2} e^{-\frac{1}{2} c f^T (L+n^{-2}I)^q f} c^{a-1} e^{-bc} \propto c^{a+n/2-1} e^{-c(b + \frac{1}{2} f^T (L+n^{-2}I)^q f)}.$$

In other words, in this case we have

$$c \mid z,f \sim \Gamma\left(a + \frac{n}{2}, b + \frac{1}{2} f^T (L+n^{-2}I)^q f\right).$$

GENERALIZED GAMMA PRIOR FOR c

If the prior density for c is the generalized gamma density given by (2.2), we do not have conjugacy. We replace drawing from the exact conditional $c \mid z,f$, as done in the preceding subsection, by a Metropolis-Hastings step. To this end, we choose a proposal density $w(c', c)$. To generate a new draw for c we follow the usual steps:

- draw a proposal $c' \sim s(\cdot, c)$;
- draw an independent uniform variable V on $[0,1]$;
- if
$$V \leq \frac{h(c')\, w(c,c')}{h(c)\, w(c',c)},$$
where
$$h(c) = c^{n/2 - r/(2q) - 1} e^{-\frac{1}{2} c f^T (L+n^{-2}I)^q) f - n c^{-r/(2q)}},$$
then accept the proposal c' as new draw, else retain the old draw c.

Note that $h(c)$ is indeed proportional to $p(f \mid c)p(c)$, as required (cf. (2.6)).

We have considered different proposal distributions $w(c', c)$ in our experiments. Our experiments indicate that a random walk proposal works well.

2.2.4. OVERVIEW OF THE SAMPLING SCHEMES

For convenience we summarize our sampling scheme, which depends on the prior for c that we use.

For the generalized gamma prior (2.2) we have the following:

Algorithm 2.1 Sampling scheme when using the generalized gamma prior on c.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, initial values for $f$ and $c$.
Output: MCMC sample from the joint posterior $p(z, f, c \mid D)$.
 1: repeat
 2:     For $i = 1, \dots, n$, draw independent
            $$z_i \sim \begin{cases} N(f_i, 1), & \text{if } i \notin I^{\text{obs}}, \\ N_+(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 1, \\ N_-(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 0. \end{cases}$$
 3:     Draw
            $$f \sim N\Big((I + c(L + n^{-2}I)^q)^{-1} z, (I + c(L + n^{-2}I)^q)^{-1}\Big).$$
 4:     Draw a proposal $c' \sim w(\cdot, c)$ and a uniform $v$ on $(0, 1)$.
 5:     if
            $$v \le \left(\frac{c'}{c}\right)^{n/2 - r/(2q) - 1} e^{-\frac{1}{2}(c'-c)f^T(L+n^{-2}I)^q f - n((c')^{-r/(2q)} - c^{-r/(2q)})} \frac{w(c, c')}{w(c', c)},$$
        then
 6:         Set $c \leftarrow c'$.
 7:     else
 8:         Retain $c$.
 9:     end if
10: until you have a large enough sample.
```

For the ordinary gamma prior (2.3) the algorithm looks as follows:

Algorithm 2.2 Sampling scheme when using the ordinary gamma prior on c.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, initial values for $f$ and $c$.
Output: MCMC sample from the joint posterior $p(z, f, c|D)$.
1: repeat
2:     For $i = 1, \dots, n$, draw independent
           $z_i \sim \begin{cases} N(f_i, 1), & \text{if } i \notin I^{\text{obs}}, \\ N_+(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 1, \\ N_-(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 0. \end{cases}$
3:     Draw
           $f \sim N\Big(((I + c(L + n^{-2}I)^q)^{-1} z, (I + c(L + n^{-2}I)^q)^{-1}\Big).$
4:     Draw
           $c \sim \Gamma\left(a + \frac{n}{1}, b + \frac{1}{2} f^T (L + n^{-2}I)^q f\right).$
5: until you have a large enough sample.
```

2.3. COMPUTATIONAL ASPECTS

2.3.1. USING THE EIGENDECOMPOSITION OF THE LAPLACIAN

In every iteration of either Algorithm 2.1 or 2.2 the matrix $I + c(L + n^{-2}I)^q$ has to be inverted. Doing this naively can in general be computationally demanding, taking $O(n^3)$ computations, in particular if L is not very sparse, i.e. if G is a dense graph with many vertices with relatively large degree. To relax the computational burden it can be advantageous to change coordinates and to work relative to a basis of eigenvectors of the Laplacian matrix L.

To make this concrete, suppose we have the eigendecomposition of the Laplacian matrix $L = U\Lambda U^T$, where Λ is the diagonal matrix of eigenvalues of L and U is an orthogonal matrix of eigenvectors. Computing this decomposition has a one time cost of $O(n^3)$. We can then parametrize the model by the vector $g = U^T f$ instead of f. The corresponding equivalent formulation of (2.4) is given by

$$\begin{aligned} D &= \{(i, y_i) : i \in I^{\text{obs}}\}, \\ I^{\text{obs}} &\sim \mu, \\ y_i &= 1_{z_i > 0}, \\ z \,|\, g, c &\sim N(Ug, I), \\ g \,|\, c &\sim N(0, (c(\Lambda + n^{-2}I)^q)^{-1}), \\ c &\sim p \quad \text{given by (2.2) or (2.3).} \end{aligned} \tag{2.7}$$

Making the appropriate, straightforward adaptations in the posterior computations, the sampling schemes take the following form in this parametrization:

Algorithm 2.3 Sampling scheme when using the generalized gamma prior on c, using $L = U\Lambda U^T$.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, initial values for $g$ and $c$.
Output: MCMC sample from the joint posterior $p(z, g, c \mid D)$.
 1: repeat
 2:     Compute $f \leftarrow Ug$ and for $i = 1, \dots, n$, draw independent
            $$z_i \sim \begin{cases} N(f_i, 1), & \text{if } i \notin I^{\text{obs}}, \\ N_+(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 1, \\ N_-(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 0. \end{cases}$$
 3:     For $j = 1, \dots, n$, draw independent
            $$g_j \sim N\left(\frac{z^T u^{(j)}}{1 + c(\lambda_j + 1/n^2)^q}, \frac{1}{1 + c(\lambda_j + 1/n^2)^q}\right).$$
 4:     Draw a proposal $c' \sim w(\cdot, c)$ and a uniform $v$ on $(0, 1)$.
 5:     if
            $$v \le \left(\frac{c'}{c}\right)^{n/2 - r/(2q) - 1} e^{-\frac{1}{2}(c'-c)\sum_{j=1}^n (\lambda_j + 1/n^2)^q g_j^2 - n((c')^{-r/(2q)} - c^{-r/(2q)})} \frac{w(c, c')}{w(c', c)},$$
        then
 6:         Set $c \leftarrow c'$.
 7:     else
 8:         Retain $c$.
 9:     end if
10: until you have a large enough sample.
```

For the ordinary gamma prior (2.3) the algorithm looks as follows:

Algorithm 2.4 Sampling scheme when using the ordinary gamma prior on c, using $L = U\Lambda U^T$.

```
Input: Data $D = \{(i, y_i) : i \in I^{obs}\}$, initial values for $g$ and $c$.
Output: MCMC sample from the joint posterior $p(z, g, c \mid D)$.
1: repeat
2:     Compute $f \leftarrow Ug$ and for $i = 1, \dots, n$, draw independent
           $z_i \sim \begin{cases} N(f_i, 1), & \text{if } i \notin I^{obs}, \\ N_+(f_i, 1), & \text{if } i \in I^{obs} \text{ and } y_i = 1, \\ N_-(f_i, 1), & \text{if } i \in I^{obs} \text{ and } y_i = 0. \end{cases}$
3:     For $j = 1, \dots, n$, draw independent
           $g_j \sim N\left(\frac{z^T u^{(j)}}{1 + c(\lambda_j + 1/n^2)^q}, \frac{1}{1 + c(\lambda_j + 1/n^2)^q}\right).$
4:     Draw
           $c \sim \Gamma\left(a + \frac{n}{2}, b + \frac{1}{2}\sum_{j=1}^{n}(\lambda_j + 1/n^2)^q g_j^2\right).$
5: until you have a large enough sample.
```

We note that in this approach, we need to invest once in the computation of the spectral decomposition of the Laplacian matrix L. This removes the $O(n^3)$ matrix inversions in each iteration of the algorithms. We do remark however that there is a matrix multiplication Ug in line 2 of the algorithms. This is in principle an $O(n^2)$ operation, which is the most expensive operation in each iteration. Moreover, Algorithms 2.3 and 2.4 produce samples of $g \mid D$, that is, we obtain the posterior samples as vectors of coordinates relative to the eigenbasis of the Laplacian matrix. If we want the samples in the original basis, which is what we need for prediction, we need to multiply all these vectors by U.

2.3.2. A STRATEGY FOR SPARSE GRAPHS

Instead of sampling the Gaussian distribution in $f \mid c, z$ at once as in (2.5), it can be advantageous to sample this vector one coordinate at the time.

Denoting by v_{-i} the vector v with coordinate i removed, standard Gaussian computations show that for every coordinate i,

$$f_i \mid f_{-i}, c, z \sim N(\mu_i, \sigma_i^2),$$

where

$$\mu_i = \frac{z_i - c((L + n^{-2}I)^q)_{i,-i} f_{-i}}{c((L + n^{-2}I)^q)_{i,i} + 1},$$

and

$$\sigma_i^2 = \frac{1}{c((L+n^{-2}I)^q)_{i,i}+1}.$$

This method for sampling from the conditional $f \mid c, z$ does not require the eigendecomposition of L. In case the power of the Laplacian matrix q is an integer, the computations for a fixed i only involve the q-step neighbors of vertex i, which might be computationally attractive in graphs where the number of q-step neighbors of each vertex is low. For example, if the number of q-step neighbors is bounded by K, the complexity of each iteration is $O(Kn)$.

2.4. NUMERICAL EXPERIMENTS

2.4.1. PATH GRAPH

To explore some of the issues involved in implementing Bayesian prediction prediction on graphs we first consider the basic example of simulated data on the path graph with $n = 500$ vertices. In this case, it is known that the Laplacian eigenvalues are $\lambda_j = 4\sin^2(\pi j/(2n))$ for $j > 1$, with corresponding eigenvectors

$$u_i^{(j)} = \frac{\sqrt{2}}{\sqrt{n}}\cos\left(\frac{\pi(i-\frac{1}{2})j}{n}\right), \quad i = 1, \ldots, n.$$

This graph satisfies the geometry condition (2.1) with $r = 1$. To simulate data we construct a function f_0 on the graph by setting

$$f_0 = \sum_{j=1}^{n} a_j u^{(j)},$$

where we choose $a_j = \sqrt{n}(j-1)^{-1.5}\sin(j-1)$ for $j > 1$ and $a_1 = 0$. This function has Sobolev-type smoothness $\beta = 1$, as defined precisely in Kirichenko and Van Zanten (2017). We simulate noisy labels y_i on the graph vertices satisfying $P(y_i = 1) = \ell_0(i) = \Phi(f_0(i))$, where Φ is the standard normal cdf. Finally we remove 20% of the labels at random to generate the set of observed labels Y^{obs}. The left panel of Figure 2.1 shows the soft label function ℓ_0 and simulated noisy hard labels Y_i on the path graph with $n = 500$ vertices. In the right panel $\log\lambda_j$ is plotted against $\log(j/n)$ to illustrate that for this graph the geometry condition indeed holds with $r = 1$.

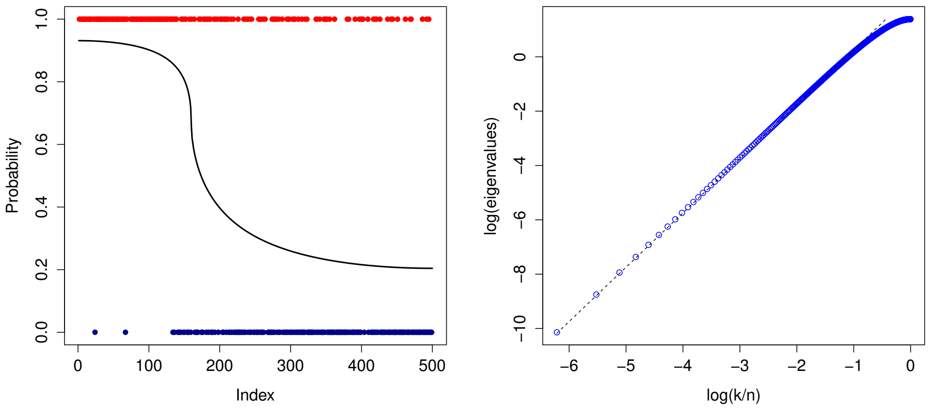


Figure 2.1: Left: soft label function and simulated noisy hard labels on a path graph with 500 nodes. Right: spectrum of the Laplacian.

In Figure 2.2, we visualize the posterior for the soft label function ℓ for various graph sizes n. Here, we used the generalized gamma prior on c with $r = 1$, $\alpha = \beta = 1$ and $q = \alpha + r/2$. These values are suggested by the theory in Kirichenko and Van Zanten (2017). The blue line is the posterior mean and the gray area depicts point-wise 95% credible intervals.

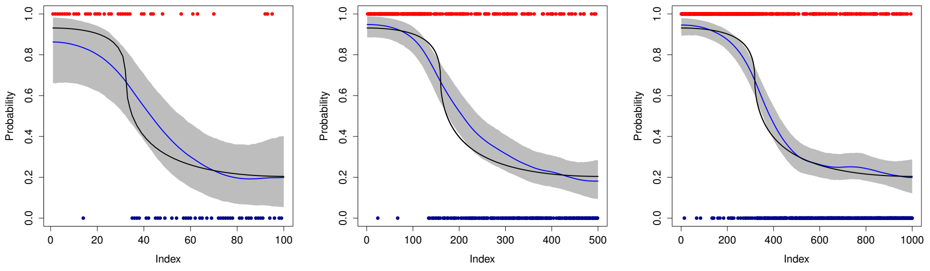


Figure 2.2: Posteriors for the soft label function for $n = 100, 500, 1000$. Prior on c is the generalized gamma.

At a first glance it appears that the procedure might be slightly oversmoothing, which could be due to the fact that the posterior for c is concentrated at too large values. To get more insight into this issue, we compare to posteriors computed with a fixed tuning parameter c, set at the oracle value which minimizes the MSE of the posterior mean, which we determined numerically. The results are given in Figure 2.3. The posteriors have slightly better coverage than those in Figure 2.2.

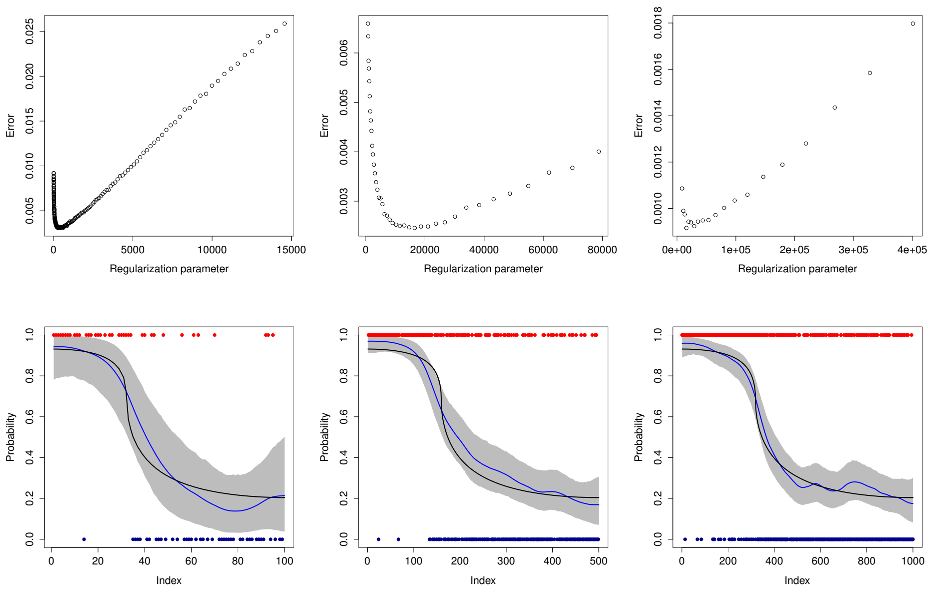


Figure 2.3: Posteriors for fixed tuning parameter c for $n = 100, 500, 1000$. Top panel: MSE of the posterior mean as function of c. Bottom panel: posterior for the soft label function for optimal choice of c. Values of c were $4.0 \cdot 10^2$, $1.6 \cdot 10^4$ and $2.9 \cdot 10^4$, respectively.

Posterior histograms of the tuning parameter show if we use the generalized gamma prior for c, the posterior indeed favours too high values of c, compared to the oracle choice. This results in the oversmoothing we observe in Figure 2.2. See the first two rows of Figure 2.4.

When instead of the theoretically optimal generalized gamma prior on c, we use the ordinary gamma prior, we can use the hyperparameters a and b to ensure that the posterior for c assigns more mass close to the oracle tuning parameter. In practice, we do not know the true underlying function, so it is natural to spread the prior mass as much as possible. We can for example choose $a = b = 0$, corresponding to an improper prior $p(c) \propto 1/c$ (as in Choudhuri et al. (2007)), or $a = 1$ and $b = 0$ such that $p(c) \propto 1$. In Figure 2.5, we plot the ordinary gamma prior density corresponding to $a = b = 0$ (blue dashed line) and the generalized gamma prior densities for various n (black lines). Since the ordinary gamma assigns more mass to smaller values of c, we might hope that if we use that prior on c, we get a posterior closer to the oracle and hence reduce the oversmoothing problem.

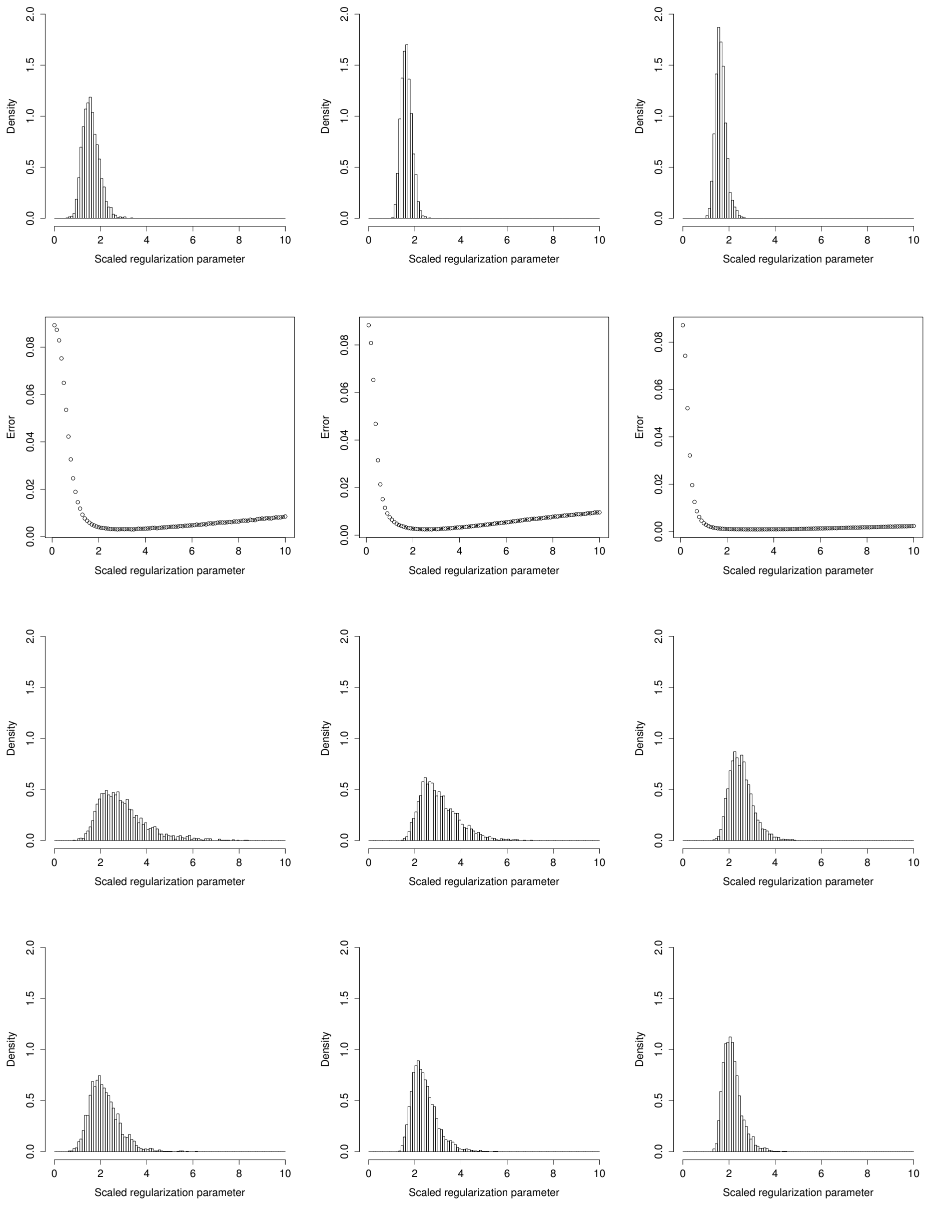


Figure 2.4: Histograms of the scaled regularization parameter $n^{1-r/(2q)}c^{-r/(2q)}$ for $n = 100, 500, 1000$. The top row: posterior corresponding to generalised gamma prior. Bottom rows: posterior corresponding to ordinary gamma prior with $a = b = 0$ and $a = 1$, $b = 0$, respectively. The second row shows the MSE of the posterior mean corresponding to fixed c as function of c.

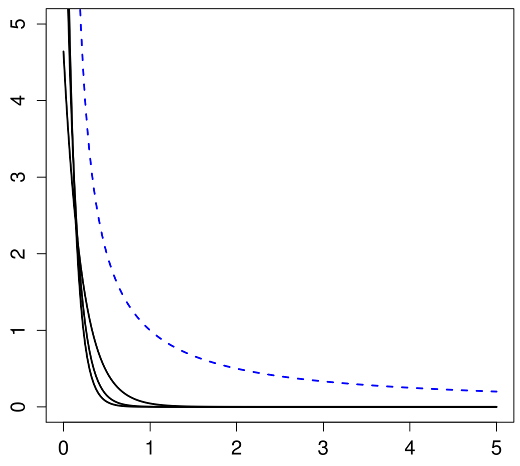


Figure 2.5: Densities of the generalized gamma prior for $n = 100, 500, 1000$ in black. Blue dashed line the ordinary gamma prior with $a = b = 0$.

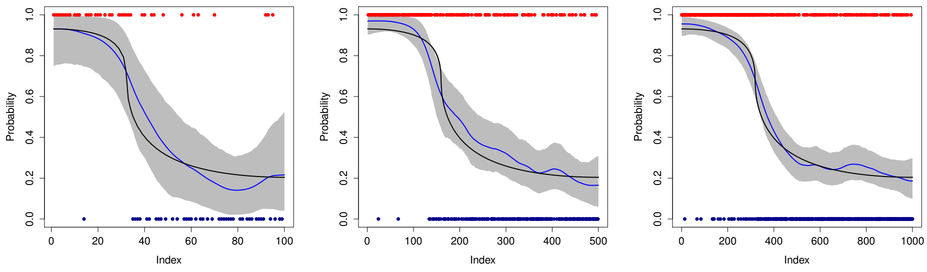


Figure 2.6: Posteriors for the soft label function for $n = 100, 500, 1000$. Prior on c is the ordinary gamma with $a = b = 0$.

Figure 2.6 visualizes the posteriors that we get for the soft label function when using the ordinary gamma prior with $a = b = 0$. We see that indeed we get better posterior coverage than in Figure 2.2. The third row in Figure 2.4 confirms that when using the ordinary gamma prior on c, the posterior for c puts more mass around the optimal value.

IMPACT OF PRIOR SMOOTHNESS

In the paper Kirichenko and Van Zanten (2017) it was suggested to take the power of the Laplacian equal to $q = \alpha + r/2$, where r is the number appearing in the geometry condition (2.1) and α is a tuning parameter that quantifies the smoothness of the prior in some sense. It was proved that when combining this with the generalized gamma prior (2.2) on

c, we get good convergence rates if the Sobolev-type smoothness of the true soft-label function is less than q. This might suggest that it is advantageous to set q high, since then the theory says that we get good rates across a large range of regularities of the true function. On the other hand, setting q higher means we favour smooth functions more. This could potentially lead to oversmoothing and hence to poor posterior coverage. In this section we investigate this issue numerically.

In Figure 2.7 we use the generalized gamma prior on c. We plot the posterior for ℓ, varying n from left to right and q from top to bottom. In the top row $q = 0.2 + r/2$. Since this is less than $\beta = 1$, the theory suggests that we are undersmoothing too much and will get sub-optimal convergence rates. The figure seems to confirms this. In the middle row we have $q = 1 + r/2$. This means the prior smoothness matches the true smoothness, which is asymptotically a good choice according to the theory. This is the same picture as in Figure 2.2. In the bottom row $q = 5 + r/2$. In this case the prior smoothness is larger than the true smoothness $\beta = 1$. However, the theory says that we should still get a good convergence rates, because to compensate for the smoothness mismatch the posterior for c will automatically charge smaller values of c more. In the simulations, we see that the result is indeed not dramatic, but that the procedure is in actual fact oversmoothing somewhat, resulting in worse posterior coverage.

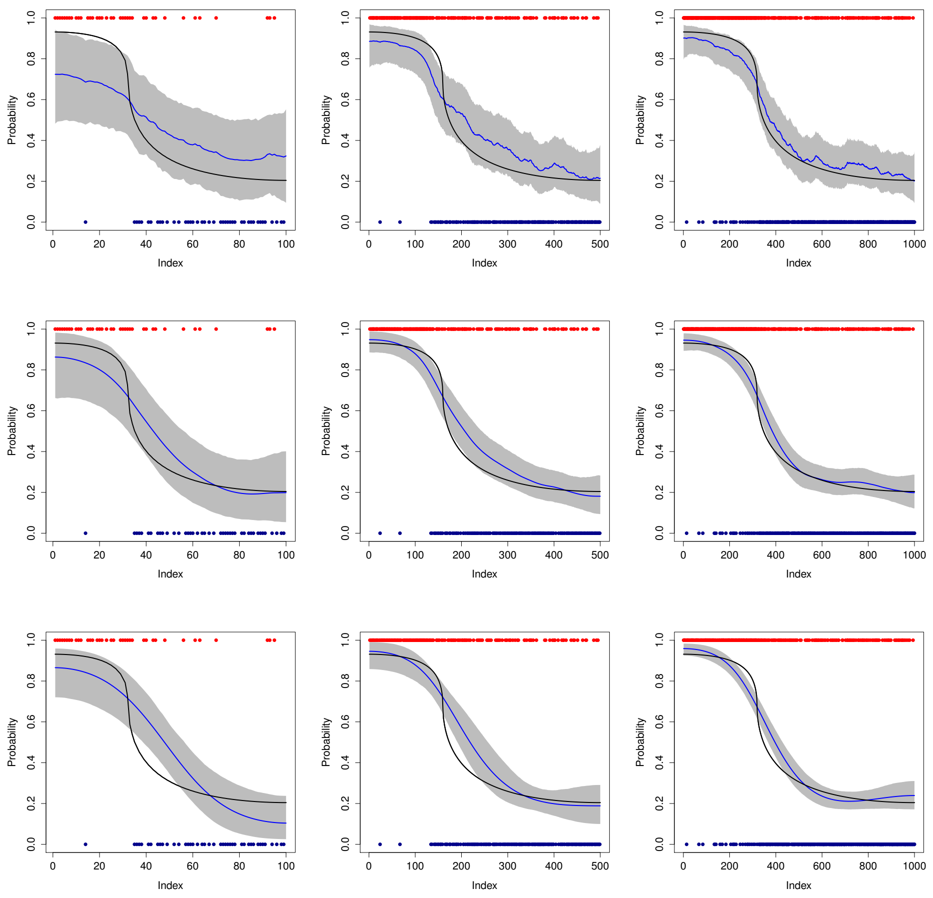


Figure 2.7: Posteriors for ℓ when using the generalized gamma prior on c. From left two right we have $n = 100, 500, 1000$. From top to bottom we have $q = \alpha + 1/2$, with $\alpha = 0.2, 1, 5$.

Figure 2.8 gives the same plots when using the ordinary gamma prior on c, with $a = b = 0$. We see essentially the same effects, but the effect of choosing q too small is a bit more pronounced. In terms of posterior coverage the ordinary gamma prior does a bit better, although it gives too conservative credible sets when q is set too low.

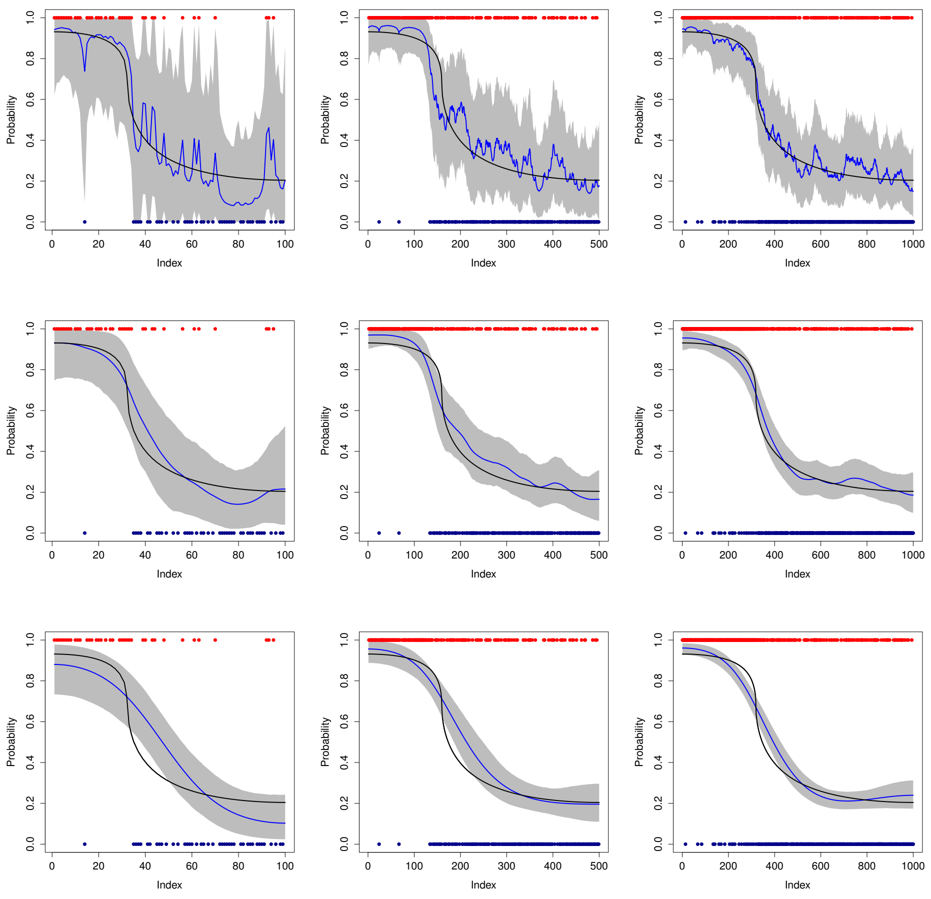


Figure 2.8: Posteriors for ℓ when using the ordinary gamma prior on c with $a = b = 0$. From left to right we have $n = 100, 500, 1000$. From top to bottom we have $q = \alpha + 1/2$, with $\alpha = 0.2, 1$ and 5 respectively.

2.4.2. SMALL-WORLD GRAPH

In this section, we consider simulated data on a small-world graph obtained as a realization of the Watts-Strogatz model (Watts and Strogatz (1998)). The graph is obtained by first considering a ring graph of 1000 nodes. Then we loop through the nodes and uniformly rewire each edge with probability 0.25. We keep the largest connected component and delete multiple edges and loops resulting in the graph in Figure 2.9 with 848 nodes.

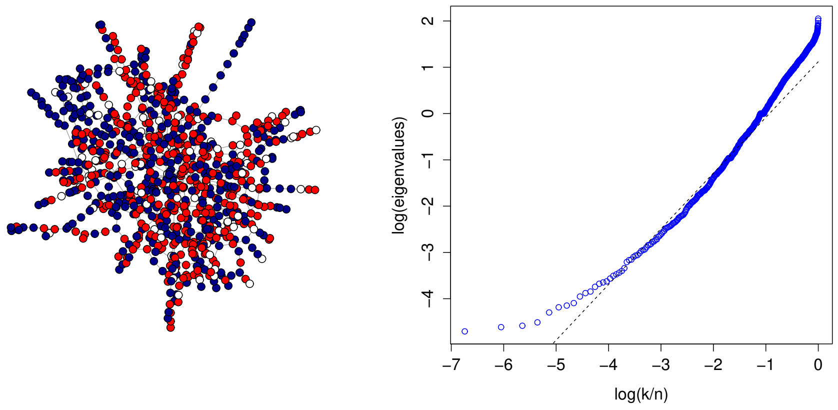


Figure 2.9: Left: small world graph with two types of labels. White labels are missing. Right: eigenvalue plot for the small-world graph. The linear fit has slope 1.20 corresponding to geometry condition with $r = 1.7$.

We numerically determine the eigenvalues λ_j and eigenfunctions $u^{(j)}$ of the Laplacian matrix and define a function f_0 on the graph by

$$f_0 = \sum_{j=1}^{n} a_j u^{(j)},$$

where we choose $a_j = \sqrt{n}(j-1)^{-2/r-1/2}\sin(j-1)$ for $j > 1$ and $a_1 = 0$ to have Sobolev-type smoothness $\beta = 2$ (cf. Kirichenko and Van Zanten (2017)). As before, we assign labels to the graph according to probabilities $P(Y_i = 1) = \ell_0(i) = \Phi(f_0(i))$, where Φ is the distribution function of the standard normal distribution. We remove the label of 10% of the nodes. The aim is to predict these using the observed labels.

In this case it is hard to visualise smooth functions on the graph and hence to visualize the entire posterior distribution of the soft label function. Instead, we analyze the quality of the procedure by plotting 95% credible intervals for the soft label function at 20 randomly selected vertices of which we have not observed the noisy labels. Figure 2.10 gives these plots for the procedure with the generalized gamma prior on c (left), the ordinary gamma prior on c with $a = b = 0$ (middle), and the fixed oracle choice of c (right). At the bottom left and middle the posteriors for c are shown. The bottom right is a plot of the MSE of the posterior mean corresponding to a fixed c as a function of that c. The point where it is minimal is the oracle choice of c.

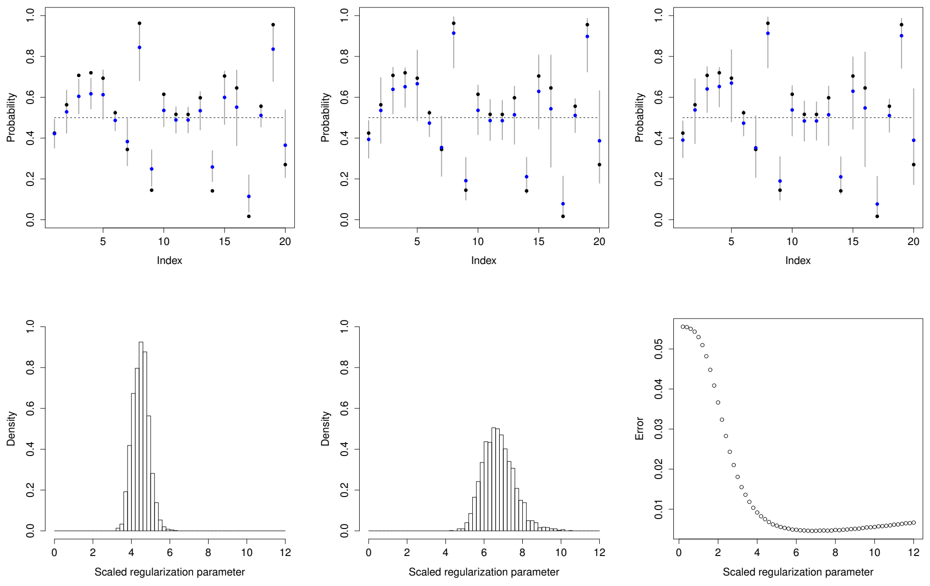


Figure 2.10: Top row: credible intervals for soft label function at 20 vertices with missing labels. Blue dots are the posterior means, black dots are the true function values. From left to right the plots correspond to the procedure with the generalized gamma prior on c, the ordinary gamma prior with $a = b = 0$, and with the fixed, oracle choice of c, respectively. Bottom row: histograms of posterior for c (scaled). The bottom right plot shows the error for different values of the scaled regularization parameter, the oracle choice of c corresponds to the value where the error is minimal.

Also in this example we observe that with the theoretically optimal generalized gamma prior on c we are shrinking a bit too much, resulting in particular in credible intervals not containing the true soft label. When using the ordinary gamma prior the performance is closer to the oracle procedure. The bottom row of Figure 2.10 confirms that with the ordinary gamma prior the posterior for c is closer to the oracle choice.

IMPACT OF HYPERPARAMETERS

We have determined numerically that the graph under consideration satisfies the geometry condition (2.1) with $r = 1.7$, see the right panel of Figure 2.9. The results of Kirichenko and Van Zanten (2017) thus suggest to use as prior on f the Laplacian to the power $q = \alpha + 1.7/2$ for parameter α that determines the prior smoothness. Also for this example we have investigated the impact of different choices.

Figure 2.11 illustrates what happens if r is chosen too low. On the left, we see that the procedure with the generalized gamma prior on c oversmooths quite dramatically. The bottom row of the figure shows that the posterior for c puts too little mass around the oracle c in that case. The plots in the middle corresponds to ordinary gamma prior on c with $a = b = 0$. This performs much better, close to procedure with oracle choice of c shown on the right. In Figure 2.12 the parameter r is chosen too high. Here all three procedures have comparable performance. All oversmoothe a bit too much due to the fact that the power of the Laplacian matrix becomes too large. This is in line with what we saw for the path graph.

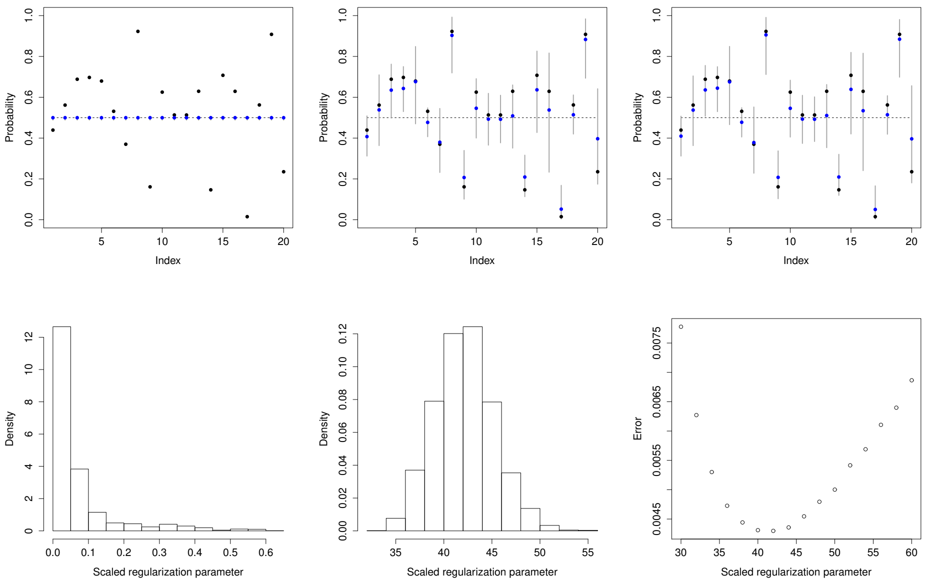


Figure 2.11: Same as Figure 2.10, but now with $r = 1$.

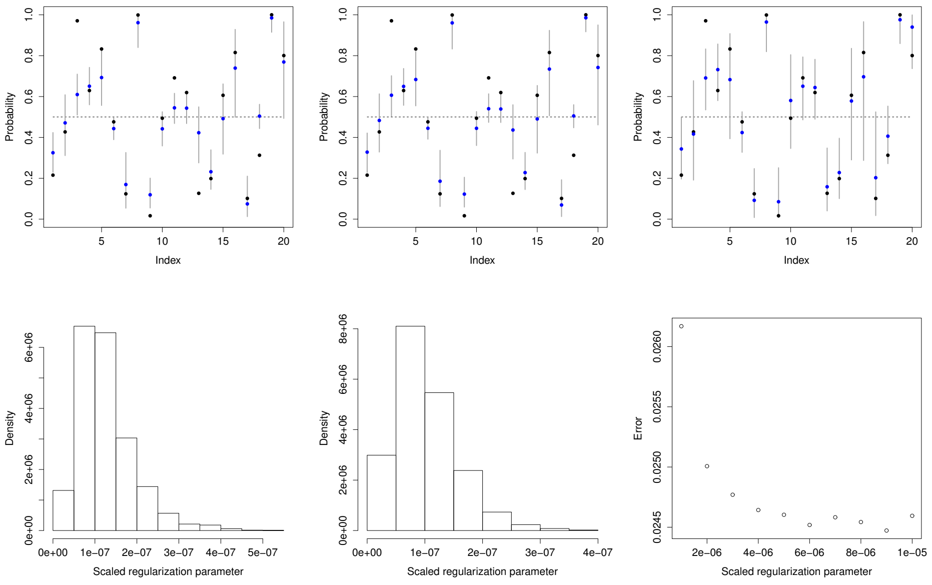


Figure 2.12: Same as Figure 2.10, but now with $r = 10$.

If we choose the parameter r correctly, i.e. $r = 1.7$ in this case, and only choose different values for the hyperparameter α we only change the prior smoothness of f, without changing the prior on c. Figures 2.13 and 2.14 illustrate the effect. In the first experiment, we set $\alpha = 0.4$, which is too low relative to the smoothness of the true soft label function, in the second one $\alpha = 10$, which is too high. We clearly see the effect on the width of the credible intervals. The effect on coverage is not very large.

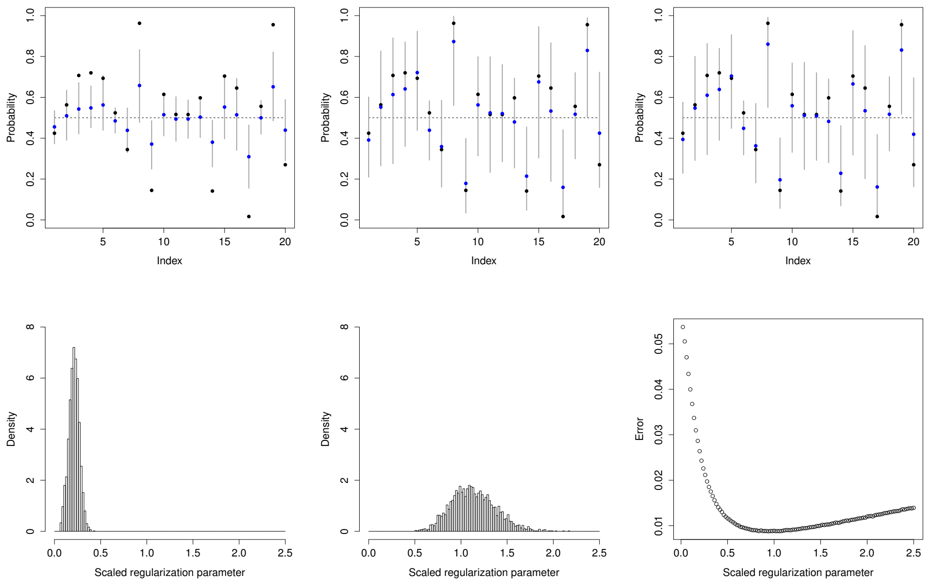


Figure 2.13: Same as Figure 2.10, but now with $\alpha = 0.4$.

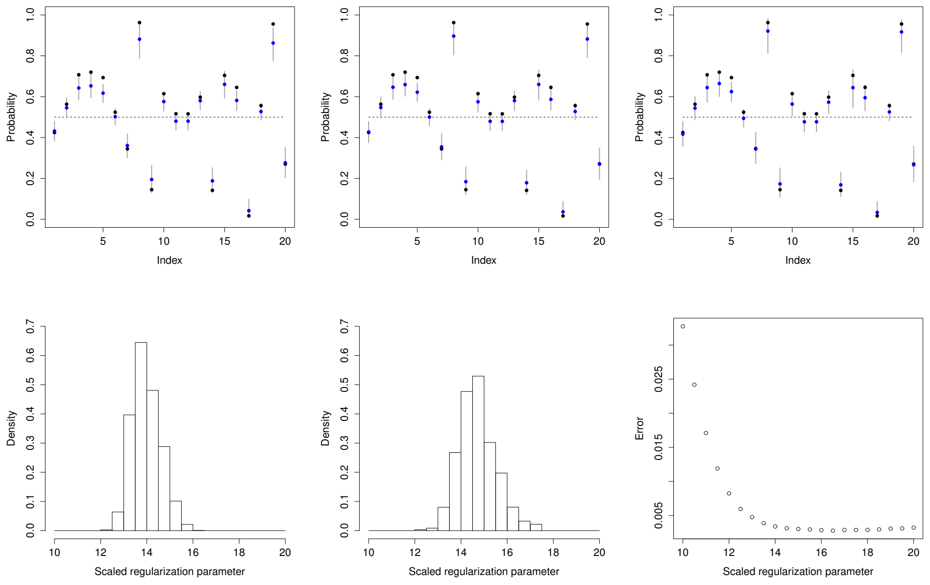


Figure 2.14: Same as Figure 2.10, but now with $\alpha = 10$.

Impact of missing observations

To assess the impact of the percentage of missing observations we increase the level from 10% to 20%, 30% and 70%. We observe in Figures 2.15, 2.16 and 2.17 that as the percentage of missing observations increases, the posterior of c is more spread out and there is more uncertainty in the function estimates, as is to be expected. In the most extreme case of 70% missing labels we observe that the generalized gamma prior on c results in quite severe oversmoothing. In the histogram we see that the posterior for c puts too little mass around the oracle choice of c in that case. The inverse gamma prior has a much better performance, and remains comparable to the oracle choice.

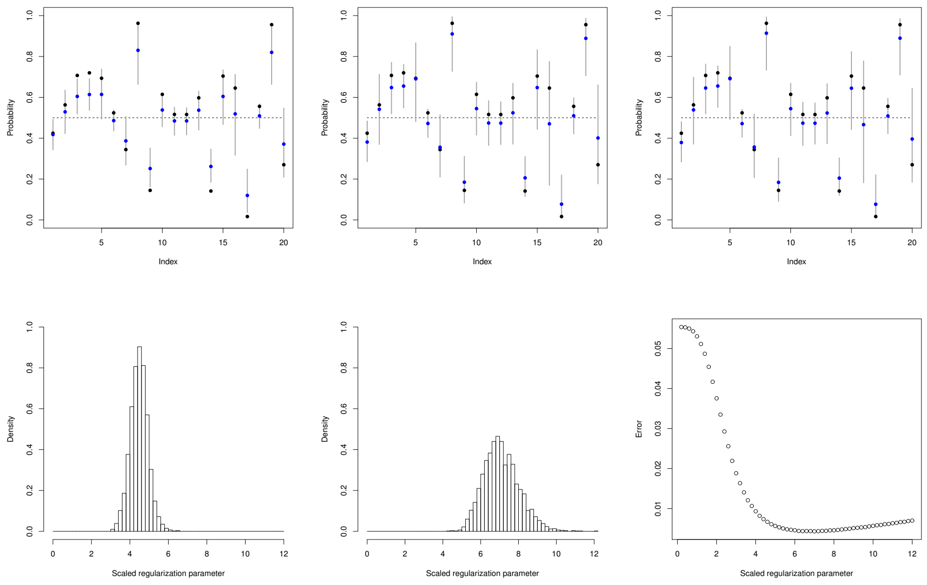


Figure 2.15: Same as Figure 2.10, but now with 20% of the label unobserved.

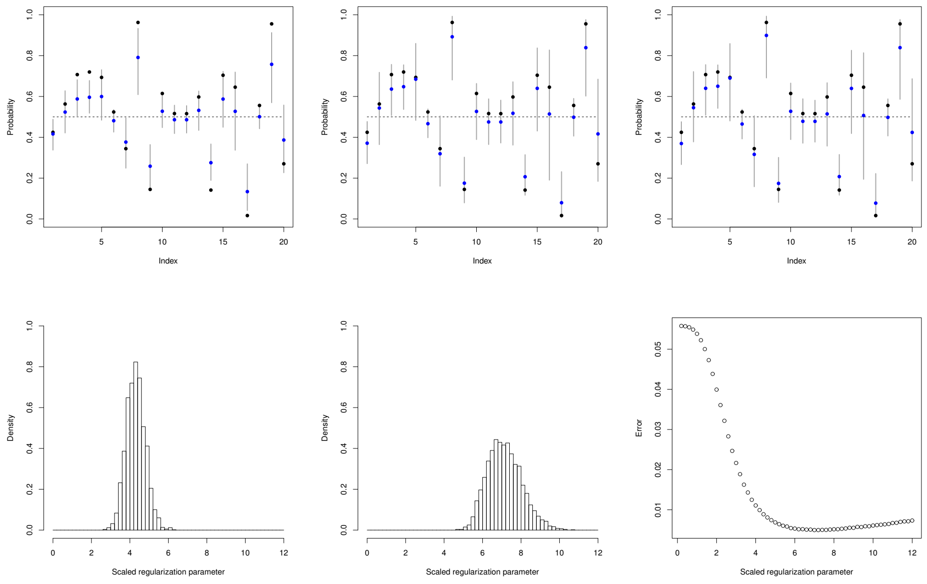


Figure 2.16: Same as Figure 2.10, but now with 30% of the label unobserved.

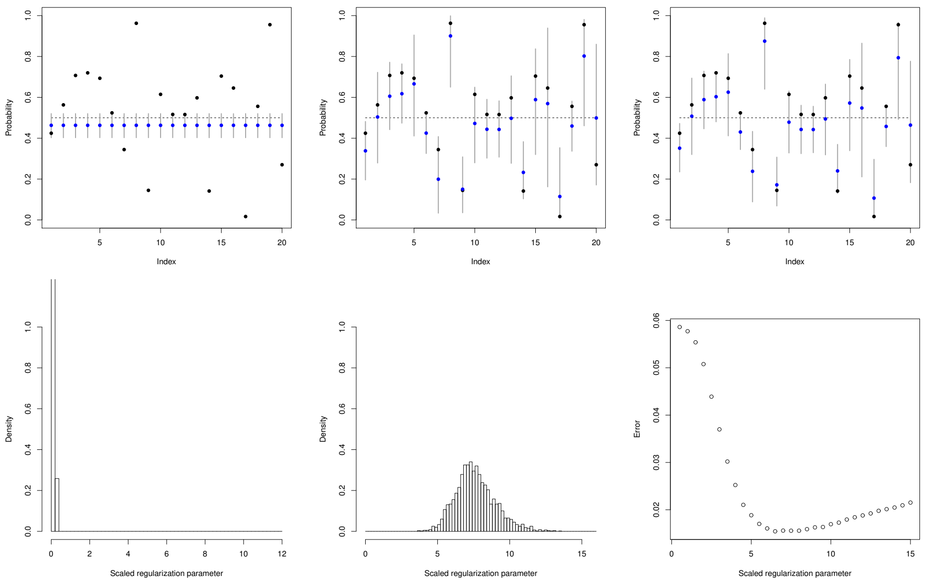


Figure 2.17: Same as Figure 2.10, but now with 70% of the label unobserved.

2.4.3. PROTEIN FUNCTION PREDICTION

To test the nonparametric Bayesian procedure on real data, we adapt the case study from Kolaczyk (2009), Section 8.5. The example is about the prediction of protein function from a network of interactions among proteins that are responsible for cell communication in yeast. For more information about the background of the experiment setup, see Kolaczyk (2009).

The protein interaction graph is shown on the left in Figure 2.18. A vertex in the graph is labelled according to whether or not the corresponding protein is involved in so-called intracellular signaling cascades (ICSC), which is a specific form of communication. We have randomly removed 12 of the labels and we apply our Bayesian prediction procedure to try and recover them from the observed labels.

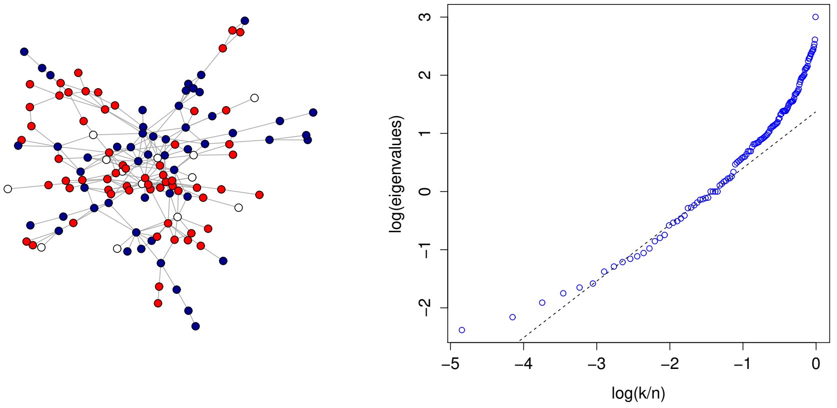


Figure 2.18: Left: protein-protein interaction graph. The red nodes in the graph are involved in ICSC, the blue nodes are not involved and the white nodes are unknown. Right: Laplacian eigenvalues. The linear fit has slope 0.97 corresponding to geometry condition with $r = 2.1$.

In view of the findings in the preceding sections we apply the procedure with the ordinary gamma prior on c with $a = b = 0$. Numerical computation of the Laplacian eigenvalues shows that in this case the geometry condition is fulfilled with $r = 2.1$, see the right panel of Figure 2.18. We use this value in the procedure. The parameter α that determines the prior smoothness of f is set to $\alpha = 1$. This is a conservative choice, in order to avoid oversmoothing. We now run our algorithm to produce credible intervals for the soft label function ℓ at the vertices of which we did not observe the labels.

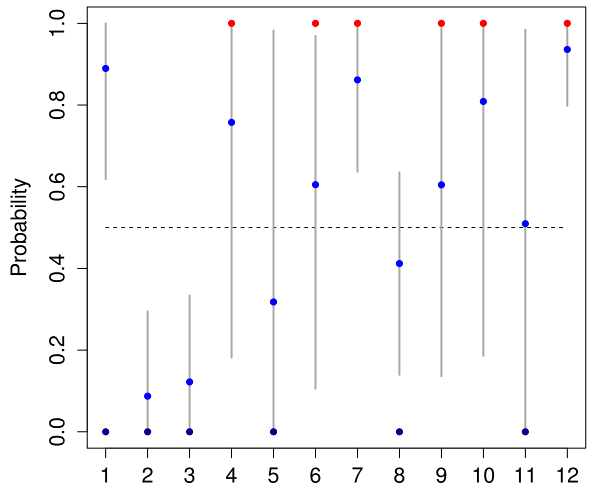


Figure 2.19: Credible intervals for the soft threshold function at the nodes with missing labels. Blue dots are the posterior means, black and red dots are the true labels. In this example, we observe a misclassification at threshold 0.5 in vertices 1 and 11.

Figure 2.19 shows the results, together with the true labels that we removed. We see that if we predict the missing labels by thresholding the posterior means at 1/2, we have a misclassification rate of 2/12 ≈ 16.7%. If we repeat the procedure 100 times, every time removing 12 different labels at random, we obtain an average misclassification rate of 27%. To assess this, we also computed k-nearest neighbour (k-NN) predictions for various k. We found average missclassification rates of 32% for one 1-NN, 28% for 2-NN and 41% for 3-NN. Hence in terms of prediction performance our procedure is comparable to k-NN with the oracle choice of k. This illustrates that, in line with the theory, our procedure succeeds in automatically tuning the appropriate degree of smoothing. Moreover, the Bayesian procedure has the advantage that in addition to predictions, we obtain credible intervals as an indication of the uncertainty in the predictions.

2.4.4. MNIST DIGIT PREDICTION

So far, we have considered examples with graphs that satisfy the geometry condition (2.1). For such graph we have theoretical results that provide some guidelines for the construction of the prior and choices of the hyperparameters. In principle however, we can also apply our procedure to graphs that do not satisfy (2.1) for some r. It is intuitively clear that a condition like (2.1) should not always be necessary for good performance. If we use the ordinary gamma prior on c and set q at a conservative (not too high) value, we can just apply our procedure and should still get reasonable results if the graph geometry and the distribution of the labels are sufficiently related. In this section, we briefly investigate such a case.

The MNIST dataset of handwritten digits has a training set of 60 thousand examples and a test set of 10 thousand. The digits are size-normalized and centered in a fixed-size image. The dataset is publicly available at `http://yann.lecun.com/exbd/mnist`. We have randomly selected a subsample of 700 consisting of only the digits 0 and 1 of which 600 in are the training set and 100 are from the test set. Our goal is to classify the 100 images from the test set. To turn this into a label prediction problem on a graph, we construct a graph with 700 nodes, corresponding to the images. For each image we determined the 10 closest images in terms of pixel-distance and connect the corresponding nodes in the graph by an edge. The resulting graph is shown in Figure 2.20. The eigenvalue plot suggests that the graph does not satisfy the geometry condition to the extent that the preceding graphs did.

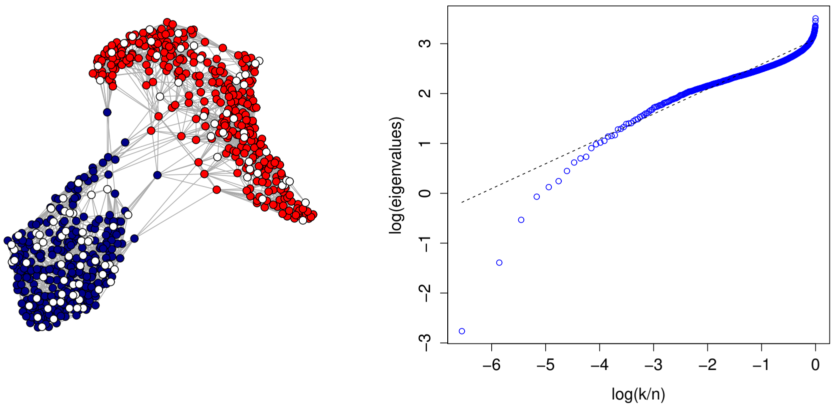


Figure 2.20: Left: constructed MNIST graph. Digits 0 are labeled blue, digits 1 are labeled red and the missing labels are white. Right: Laplacian eigenvalues.

The picture indicates that predicting the missing labels in this graph is not a very hard problem. And indeed, our procedure performs well in this case. We classify all missing labels correctly, with very high certainties. See Figure 2.21.

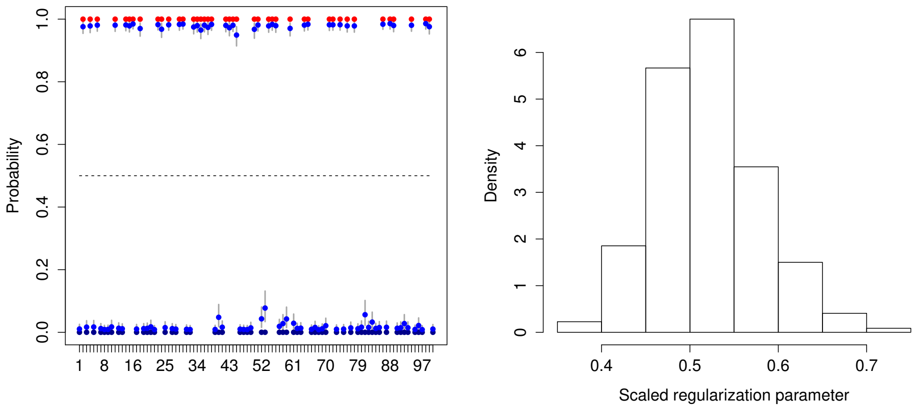


Figure 2.21: Left: credible intervals for the soft threshold function at the nodes with missing labels. Blue dots are the posterior means, black and red dots are the true labels. Right: posterior for tuning parameter c.

2.5. Regression problem

In this section, we demonstrate a hierarchical Bayesian approach to solving the regression problem on a graph. The graph is denoted by $G = (V, E)$, where $V = \{1, \dots, n\}$. We assume that the noisy labels y are Gaussian with

$$y \,|\, f \sim N(f, \sigma^2 I),$$

where f is the function of interest. For simplicity, we assume σ^2 is fixed and known and that all noisy label are observed. We are interested in inferring f from the data y. Similar to the classification problem in the previous sections, we consider a Gaussian prior on f, which depends on the Laplacian matrix L and on a multiplicative scaling parameter $c > 0$. The resulting prior is

$$f \,|\, c \sim N(0, (c(L + n^{-2} I))^{-q}),$$

for some fixed $q > 0$. The additional number n^{-2} is added to L to make it invertible as in Section 2.1. In this chapter, we have considered the generalized gamma prior (2.2) and the ordinary gamma prior (2.3) for the regularization parameter c. Motivated by the numerical experiments in Section 2.4, we only consider the ordinary gamma prior

$$c \sim \Gamma(a, b)$$

for $a, b > 0$. The improper case of $a = b = 0$ is also used in the literature (e.g. Choudhuri et al. (2007); Liang et al. (2007)). In Section 2.4, we have seen that this is a reasonable choice in our setting as well. The three distributions of $y \,|\, f$, $f \,|\, c$ and c form the hierarchical model.

To sample from the posterior $f \,|\, y$, we use a Gibbs sampler as in Algorithm 1.1. We can naturally partition the unknown parameter as (f, c). To sample from the joint posterior $f, c \,|\, y$ we iteratively sample from the conditionals $f \,|\, c, y$ and $c \,|\, f, y$.

We have

$$p(f\,|\,c,y) \propto p(y\,|\,f,c)p(f\,|\,c).$$

Note that $y\,|\,f$ is independent of c. We can plug in the two densities to get

$$p(f\,|\,c,y) \propto e^{-\frac{1}{2}(y-f)^T(y-f)/\sigma^2 - \frac{1}{2}cf^T(L+n^{-2}I)^q f}.$$

From this expression, we see the connection with kernel-based maximum likelihood methods (e.g. Kolaczyk (2009)). The first term in the log posterior density of $f\,|\,c,y$ is the squared error of our estimate f with the observed data y. The second term is the kernel-based norm of f. The parameter c balances smoothness in terms of the kernel-based norm with fit to the data. We can rearrange $p(f\,|\,c,y)$ to see that

$$f\,|\,c,y \sim N(\sigma^{-2}(\sigma^{-2}I + c(L+n^{-2}I)^q)^{-1}y, (\sigma^{-2}I + c(L+n^{-2}I)^q)^{-1}).$$

The conditional $p(c\,|\,f,y)$ can be found in a similar manner, this time exploiting the normal-inverse gamma partial conjugacy. It should be noted that given f, y is independent of c, so $c\,|\,f,y$ does not depend on y. We find

$$c\,|\,f,y \sim \Gamma\left(a+\frac{n}{2}, b+\frac{1}{2}f^T(L+n^{-2}I)^q f\right).$$

We see that the improper prior with $a=b=0$ results in a proper posterior.

As in Section 2.3, we can use the eigendecomposition of the Laplacian matrix $L = U\Lambda U^T$ to change coordinates to a basis of its eigenvectors. The model is parametrized by the vector $g = U^T f$ instead of f. We can make the appropriate changes to the full conditionals to find a sampling scheme for the posterior $p(g,c\,|\,D)$.

These steps can be summarized in the following algorithm:

Algorithm 2.5 Simple sampling scheme for the regression problem.

```
Input: Data $D = \{(i, y_i) : i = 1,\dots,n\}$, variance $\sigma^2$, initial value for $c$.
Output: MCMC sample from the joint posterior $g, c\,|\,y$.
1: repeat
2:     For $j = 1,\dots,n$, draw
           $g_j \sim N(\sigma^{-2}(\sigma^{-2} + c(\lambda_j + n^{-2})^q)^{-1}(U^T y)_j, (\sigma^{-2} + c(\lambda_j + n^{-2})^q)^{-1})$.
3:     Draw
           $c \sim \Gamma\left(a+\frac{n}{2}, b+\frac{1}{2}\sum_{j=1}^{n}(\sigma^{-2} + c(\lambda_j + n^{-2})^q)g_j^2\right)$.
4: until you have a large enough sample.
```

When σ^2 is unknown, we can put an independent prior $p(\sigma^2)$ on σ^2, or rather, an independent prior $p(\sigma^{-2})$ on the precision σ^{-2}. We include a draw from $\sigma^{-2}\,|\,c,g,D$ in

Algorithm 2.5 to sample from the joint posterior $g, c, \sigma^{-2} \mid D$. If we take a gamma prior for σ^{-2} with shape s and rate t, we see that

$$\sigma^{-2} \mid c, g, D \sim \Gamma\left(s + \frac{n}{2}, t + \frac{1}{2}\sum_{i=1}^{n} y_i^2 + \frac{1}{2}\sum_{j=1}^{n}(g_j^2 - 2g_j(U^T y)_j)\right),$$

where we have written the quadratic form $(y - Ug)^T(y - Ug)$ in such a way that we avoid the matrix-vector multiplication Ug iteration, but only require $U^T y$, which can be computed beforehand. Note that this uses the orthogonality $U^T U = I$.

To account for missing data, we can include the missing observations y^{miss} as latent variables and impute them prior to line 2 of Algorithm 2.5 in every iteration using

$$y^{\text{miss}} \mid c, g, \sigma^{-2}, D \sim N(U^{\text{miss}} g, (1/\sigma^{-2})I),$$

where U^{miss} is the restriction of U to the rows corresponding to the missing observations. The other steps in Algorithm 2.5 remain unchanged as we simply use $y = (y^{\text{obs}}, y^{\text{miss}})$. A drawback is that $U^T y$ can no longer be computed beforehand, but has to be updated in every iteration. However, we can decompose $U^T y = (U^{\text{obs}})^T y^{\text{obs}} + (U^{\text{miss}})^T y^{\text{miss}}$ and only update the part corresponding to the missing values. A sampling scheme for the regression problem with missing data and unknown noise level is Algorithm 2.6.

Algorithm 2.6 Sampling scheme for the regression problem.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, initial values for $c$, $g$ and $\sigma^{-2}$.
Output: MCMC sample from the joint posterior $y^{\text{miss}}, c, g, \sigma^{-2} \mid D$.
1: repeat
2:     For $i \notin I^{\text{obs}}$, draw
           $y_i \sim N(U_{i,\cdot} g, 1/\sigma^{-2})$.
3:     For $j = 1, \ldots, n$, draw
           $g_j \sim N(\sigma^{-2}(\sigma^{-2} + c(\lambda_j + n^{-2})^q)^{-1}(U^T y)_j, (\sigma^{-2} + c(\lambda_j + n^{-2})^q)^{-1})$.
4:     Draw
           $c \sim \Gamma\left(a + \frac{n}{2}, b + \frac{1}{2}\sum_{j=1}^{n}(\sigma^{-2} + c(\lambda_j + n^{-2})^q) g_j^2\right)$.
5:     Draw
           $\sigma^{-2} \sim \Gamma\left(s + \frac{n}{2}, t + \frac{1}{2}\sum_{i=1}^{n} y_i^2 + \frac{1}{2}\sum_{j=1}^{n}(g_j^2 - 2g_j(U^T y)_j)\right)$.
6: until you have a large enough sample.
```

2.5.1. TRAFFIC FLOW ESTIMATION

To illustrate the regression problem, we consider a simulated example of traffic flow estimation inspired by Aldous and Shun (2010) and Bickel et al. (2007). Traffic data is collected by authorities to measure the performance of the road network in terms of flow, occupancy and speed at locations in the network. A common type of sensor used for traffic flow estimation is a *loop detector* that counts the number of passing vehicles over a certain time interval. However, the resulting loop data is often missing or invalid due to communication errors or malfunction. Missing and bad observations are imputed from the neighboring sensor location using a simple average or a regression approach.

We simulate a road network between cities based on a toy model. We sample 500 points uniformly on the unit square and compute the *relative neighborhood graph* by connecting nodes u and v if there does not exist a node w such that

$$\max\{\|u-w\|, \|v-w\|\} < \|u-v\|.$$

In Figure 2.22 this definition is illustrated, nodes u and v are connected if there is no node in the gray area. The resulting graph is simple and connected. Relative neighborhood graphs and related proximity graphs, such as the nearest neighbor graph in Section 2.4.4, can be used as a toy models for road networks (see Aldous and Shun (2010)).

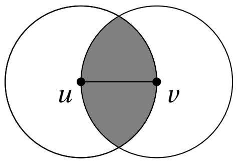


Figure 2.22: Construction of a relative neighborhood graph. Nodes u and v are connected if there is no point closer to both u and v as they are to each other, corresponding to the gray area not containing a node.

The resulting graph is the left graph in Figure 2.23. As traffic flow is a property of the roads in the city graph, we construct its *line graph*, by associating a vertex with each road in the city graph and connecting two vertices if the corresponding roads in the city graph have a city in common. This is the right graph in Figure 2.23 and is the road graph G we work with. It has $n = 617$ vertices.

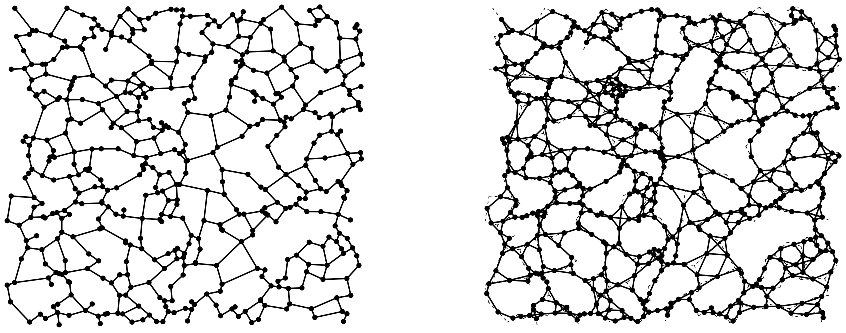


Figure 2.23: Left: A road network between cities as a realization of the relative neighborhood graph on 500 random uniform points on the unit square. Right: The corresponding line graph, where the nodes correspond to the roads of the left graph. The dashed line is the original graph.

We construct a true traffic flow function f_0 on the graph by setting

$$f_0 = \sum_{j=1}^{n} a_j u^{(j)},$$

where $u^{(j)}$ are the eigenvectors of the Laplacian matrix $L = U\Lambda U^T$ of G. To construct the coefficients a_j we consider the path graph with vertices $x_i = i/n$ for $i = 1,\ldots,n$ and the function $5 + f(x_i)$, where f is given by Equation (1.8). We compute a_j as the coefficients of $5 + f$ on the path graph and use them in our graph G, generalizing the function on the path graph to our graph G. The resulting underlying traffic flow is depicted in Figure 2.24 on both the city graph and the road graph G.

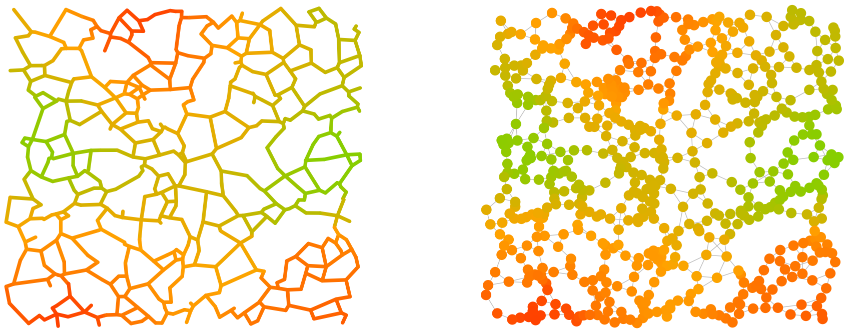


Figure 2.24: Left: The corresponding traffic flow on the city graph, where the flow is observed on the edges. Right: The traffic flow on the road graph G. Traffic intensity ranges from green (light traffic) to red (heavy traffic) and is observed on the nodes.

To construct a noisy observation of the traffic flow, we add independent noise $\epsilon_i \sim N(0,1)$ to each node $i = 1,\ldots,n$ and remove 123 of our observations as depicted in Figure 2.25.

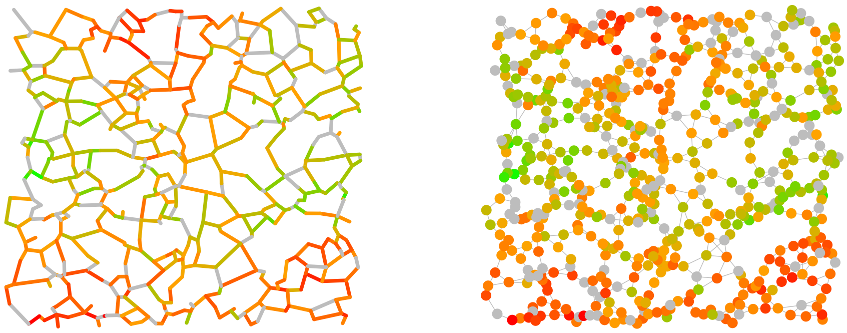


Figure 2.25: The noisy observation of the underlying traffic flow in Figure 2.24. The missing observations are in gray.

We estimate the traffic flow using Algorithm 2.6 with hyperparameters $a = b = 0$, $s = t = 1$ and $q = 15$. We numerically determine $r = 1.67$ from the spectrum of the Laplacian matrix as shown in Figure 2.26. The resulting posterior mean for $f = Ug$ is shown in Figure 2.27. To illustrate the uncertainty in our estimate, we also plot the estimated traffic flow along a path from the north-west corner of the map to the south-east corner in Figure 2.28 together with point-wise 95% credible intervals.

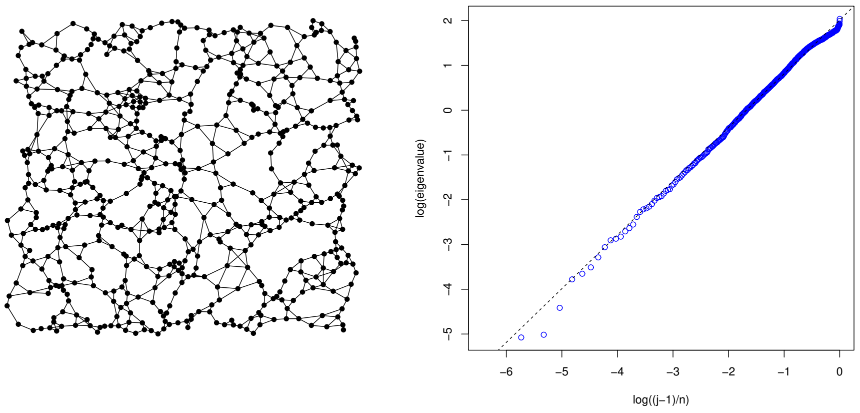


Figure 2.26: Left: The road graph. Right: The eigenvalues of the Laplacian matrix of the road graph. The dashed line corresponds to geometry number $r = 1.67$ based on a linear fit with slope 1.24. We have used the first 35% of the eigenvalues with exception of the first three.

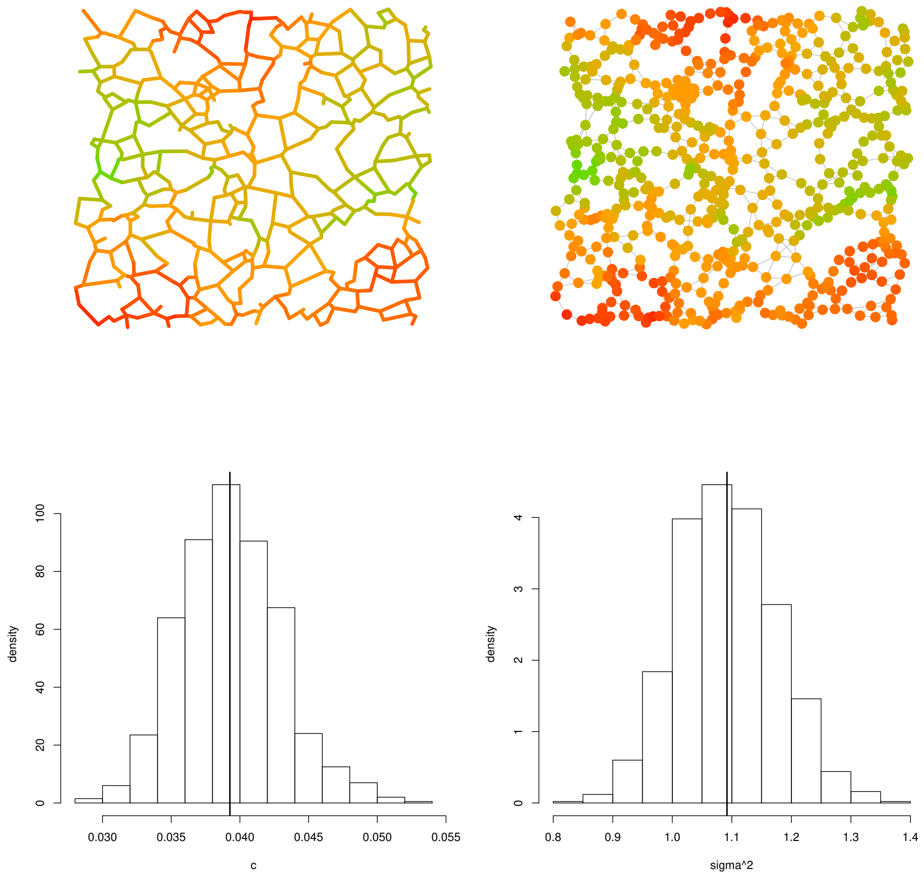


Figure 2.27: The posterior mean for the traffic flow data from Figure 2.25. Bottom: Histograms for the posterior distribution of c and σ^2. The vertical black lines are the corresponding posterior means.

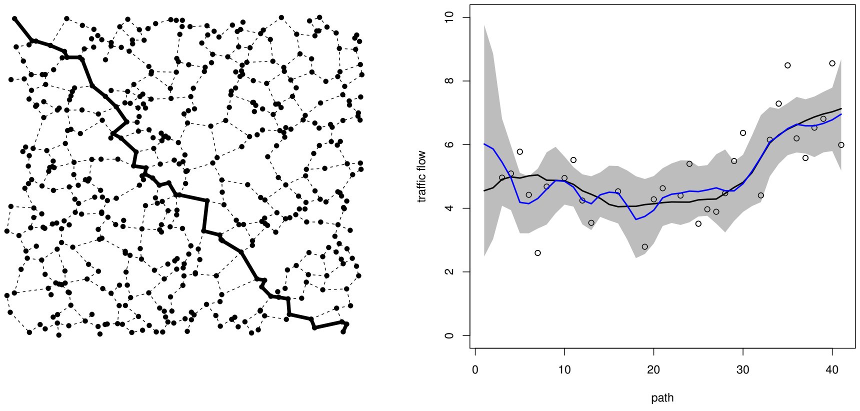


Figure 2.28: Left: A path in the city graph. Right: Posterior mean (blue) and point-wise 95% credible interval (gray area) for the traffic flow along the path. The points are the noisy observations and the black line is the true underlying traffic flow function.

2.6. CONCLUDING REMARKS

We have described nonparametric Bayesian procedures to perform regression and binary classification on graphs. We have considered a hierarchical Bayesian approach with a randomly scaled Gaussian prior as in the theoretical framework in Kirichenko and Van Zanten (2017). We have implemented the procedure with the theoretically optimal prior from Kirichenko and Van Zanten (2017) and a variant with a different prior on the scale, which exploits partial conjugacy and has some more flexibility.

Our numerical experiments suggest that good results are obtained for the classification problem, when using Algorithm 2.4, i.e. using the ordinary gamma prior on the scale. Suggested choices for the hyperparameters are $a = b = 0$ and $q = \alpha + r/2$. Here r is the geometry parameter appearing in the geometry condition (2.1) and can be determined numerically from the spectrum of the Laplacian matrix. The parameter α reflects prior smoothness and should not be set too high (e.g. $\alpha = 1$ or 2), to avoid oversmoothing. For the regression problem, we have found good results using Algorithm 2.6. In the example in Section 2.5, we have used a larger prior smoothness as the true underlying function was very smooth.

In view of computational complexity it might be more advantageous to consider other methods to adaptively find the tuning parameter, such as empirical Bayes methods (see Chapter 4). Also, it might be sensible to modify the prior by truncating the n-dimensional Gaussian prior on f to a lower dimensional one by writing a series expansion for f and truncating the sum at a random point, similar to the approach in Liang et al. (2007) for instance (see Chapter 3). It is conceivable that in this way the procedure becomes both more flexible in terms of adaptation to smoothness and will also computationally scale better to large sample size n.

3

TRUNCATION

In this chapter, we describe an implementation of a nonparametric Bayesian approach to solve regression and binary classification problems on graphs. We consider a hierarchical Bayesian approach with a prior that is constructed by truncating a series expansion of the soft label function using the graph Laplacian eigenfunctions as basis functions. We compare our truncated prior to the untruncated Laplacian-based prior in simulated and real data examples to illustrate the improved scalability in terms of size of the underlying graph.

In this chapter, we consider an extension to the method proposed in Hartog and Van Zanten (2018) (Chapter 2) in the context of regression and binary classification problems on graphs. The setup of the regression problem and the classification problem remains the same as in Chapter 2 and outlined in Section 1.4. We have noisy observations of the labels of part of the vertices of a large given graph and the goal is to classify all vertices correctly, including those for which there is no observation available. In Hartog and Van Zanten (2018) an implementation is provided of nonparametric Bayesian prediction on graphs using Gaussian priors based on the Laplacian matrix of the graph. Using the eigendecomposition of the Laplacian matrix, we can view this prior as a Gaussian series prior

$$f = \sum_{j=1}^{n} g_j u^{(j)}, \tag{3.1}$$

where n is the number of vertices of the graph, $u^{(j)}$ are the eigenvectors of the Laplacian matrix and g_j Gaussian random variables for $j = 1,\dots,n$. As indicated in Hartog and Van Zanten (2018), using the full Laplacian, i.e. all n eigenvectors, is computationally demanding and limits the applicability of a Bayesian procedure with this natural prior for very large graphs. In the present chapter, we address this issue by truncating the series at a random point for computational efficiency. This leads to a number of practical

Parts of this chapter have been published in Communications in Statistics - Simulation and Computation (2019) (Hartog and Van Zanten, 2019).

issues regarding prior choices etcetera, which we address in a simulation study. We illustrate the improved scalability by considering an example involving a graph with $90,000$ nodes.

Another advantage of truncating the series (3.1) at a random point is that it yields a more flexible prior in terms of adaptation to smoothness. Theoretical results for random inverse-gamma scaling of series priors with Gaussian coefficients and random truncation are given in Van Waaij and Van Zanten (2017) in the context of signal in white noise and estimating the drift function of a diffusion process. In these contexts it was shown that the truncated series prior with a geometric or Poisson prior on the truncation level achieves the optimal posterior contraction rate. Although in this work we are in a different setup where the results of Van Waaij and Van Zanten (2017) do not directly apply, we will also use a geometric prior and our proposed method will be a reversible jump Markov chain Monte Carlo algorithm similar to the method in Van der Meulen et al. (2014) in the context of diffusion processes.

The rest of this chapter is organized as follows. In the next section, we describe the classification problem setting and the priors we consider. A sampling scheme to draw from the posterior distribution is given in Section 3.2 and some computational aspects are discussed in Section 3.3. In Section 3.4 we present numerical experiments. We first apply our method on a simple example on the path graph to illustrate the impact of the prior on the truncation level. In the next example we use data from the MNIST dataset to illustrate how the truncated prior is more attractive than an untruncated prior in terms of computation time at a similar level of prediction accuracy. As a final example we apply our algorithm to a simple object tracking problem in a noisy environment, to further illustrate the improved scalability achieved by truncation. In Section 3.5 we show the applicability of the truncated prior to the regression problem and illustrate it with a simulated example. In Section 3.6 some concluding remarks are given.

3.1. OBSERVATIONAL MODEL AND PRIORS

3.1.1. OBSERVATIONAL MODEL

Our problem setup is the same as in Chapter 2 (Hartog and Van Zanten (2018)). We have a given connected, simple graph $G = (V, E)$, with $\#V = n$ vertices, denoted for simplicity by $V = \{1, \dots, n\}$. Associated to every vertex i is a random hard label $y_i \in \{0, 1\}$. We assume that the variables y_i are independent, so that their joint distribution is determined by the unobserved soft label function $\ell : V \to (0, 1)$ given by

$$\ell(i) = P(y_i = 1) = 1 - P(y_i = 0).$$

The observed data is $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, where $I^{\text{obs}} \subset V$ is drawn from an arbitrary distribution μ on the collection 2^V of subsets of vertices. The exact sampling mechanism μ is not important for the algorithm we propose, only that the subset is independent of the labels. Throughout, we use the well-known latent variable perspective on this model (cf. Albert and Chib (1993)). This is simply the observation that we can sample Bernoulli variables $y_1, \dots, y_n$ with success probabilities $\ell(1), \dots, \ell(n)$ using an intermediate layer of latent Gaussian variables. Indeed, let Φ be the probit link, i.e. the cdf of the standard normal distribution. Then, if $f : V \to \mathbb{R}$ is given, sampling independent Bernoulli variables y_i with success probabilities $\ell(i) = \Phi(f(i))$ can be achieved by subsequently sampling independent Gaussian variables z_i with mean $f(i)$ and variance 1 and then setting $y_i = 1_{z_i > 1}$ for $i = 1,\dots,n$.

3.1.2. PRIOR ON THE SOFT LABEL FUNCTION

The idea proposed in Hartog and Van Zanten (2018) is essentially to achieve a form of Bayesian Laplacian regularization in this problem by putting a Gaussian prior on the function f that determines the distribution of the hard labels, with a precision (inverse covariance) matrix given by a power of the Laplacian matrix L. The Laplacian matrix is given by $L = D - A$, with A the adjacency matrix of the graph and D the diagonal matrix of vertex degrees. It is a symmetric, non-negative definite matrix. Since it always has eigenvalue 0 however, it is not invertible. A slight adaptation is necessary before it can serve as a precision matrix. In Hartog and Van Zanten (2018), the matrix L is made invertible by adding a small number $1/n^2$ to the diagonal, motivated by the result that the smallest nonzero eigenvalue of the Laplacian matrix is at least $4/n^2$ (Theorem 4.2 of Mohar (1991)). Adding a multiplicative scale parameter $c > 0$ and a hyperparameter $q \geq 0$ as well, Hartog and Van Zanten (2018) propose to employ the prior

$$f \mid c \sim N(0, (c(L + n^{-2}I)^q)^{-1}).$$

Using the eigendecomposition of the Laplacian matrix $L = U\Lambda U^T$, with Λ the matrix of Laplacian eigenvalues and U the orthogonal matrix containing the corresponding eigenvectors, we can write $f = Ug$ for some vector g and write the prior proposed in Hartog and Van Zanten (2018) in series form as

$$f \mid c \sim \sum_{j=1}^{n} g_j u^{(j)},$$

where $u^{(j)}$ is the jth eigenvector of L and

$$g \mid c \sim N(0, (c(\Lambda + n^{-2}I)^q)^{-1}),$$

where we assume the eigenvalues are ordered by magnitude, i.e. $0 = \lambda_1 < \lambda_2 \leq \cdots, \leq \lambda_n$.

In this chapter, we propose a prior that is more flexible and that improves scalability with the graph size n. We truncate the above series at a random point k to be equipped with a prior distribution. Specifically, the prior we use in this chapter can then be written as

$$f \mid k, c \sim \sum_{j=1}^{k} g_j u^{(j)},$$

which depends on the random truncation level k and random scale parameter c via g. The prior on g given c and k is in this case

$$g \mid k, c \sim N(0, (c(\Lambda_k + n^{-2}I)^q)^{-1}),$$

where Λ_k denotes the restriction of Λ to the first k rows and columns.

3.1.3. PRIOR ON THE TRUNCATION LEVEL

As we wish to express some preference for small models, i.e. low values for k, we use a truncated exponential prior with rate γ with probability mass function

$$p(k) \propto e^{-\gamma k}, \quad k = 1, \dots, n. \tag{3.2}$$

The rate γ controls how strongly we prefer small models over large models, with the limiting case $\gamma \to 0$ giving uniform mass $1/n$ to all possible values $k = 1, \dots, n$. It can be seen that for every $l \in \{1, \dots, n\}$, the prior (3.2) on k assigns mass

$$(1 - e^{-\gamma l}) \frac{e^{\gamma n}}{e^{\gamma n} - 1}$$

to $\{1, \dots, l\}$. For large graphs this is approximately $1 - e^{-\gamma l}$ and this can be used to set γ in such a way that the prior is mostly concentrated on the first l eigenvectors, possibly relieving the computational burden of having to compute all the eigenvalues. In some cases this might result in oversmoothing, but for large graphs it might simply be prohibitive to compute all the eigenvectors.

In our numerical experiments ahead we use the rule-of-thumb of setting $\gamma = 20/n$, unless otherwise stated. This corresponds to concentrating the prior mass on the first eigenvectors. Specifically, for this choice it holds that approximately 63% of the prior mass is on the first 5% of the eigenvectors, 86% mass on the first 10% of the eigenvectors and 98% mass is on the first 20% of the eigenvectors. Simulations indicate that this is an appropriate choice in many situations.

3.1.4. PRIOR ON THE REGULARIZATION PARAMETER

We use the natural choice of prior for c, which is a gamma prior with density

$$p(c) \propto c^{a-1} e^{-bc}, \quad c > 0$$

for certain $a, b > 0$. This choice is motivated by the normal-inverse gamma partial conjugacy (see e.g. Choudhuri et al. (2007); Liang et al. (2007) in the context of our setting) and the positive results in the numerical experiments in Hartog and Van Zanten (2018). We can even choose the improper prior corresponding to $a = b = 0$, in which case $p(c) \propto 1/c$ (as in Choudhuri et al. (2007); Hartog and Van Zanten (2018)) or $a = 1$ and $b = 0$, such that $p(c) \propto 1$.

3.1.5. Full hierarchical model

All in all, the full hierarchical scheme considered here is the following:

$$
\begin{aligned}
D &= \{(i, y_i) : i \in I^{\text{obs}}\}, \\
I^{\text{obs}} &\sim \mu, \\
y_i &= 1_{z_i > 0}, \quad i = 1, \dots, n, \\
z \mid f &\sim N(f, I), \\
f &= \sum_{j=1}^{k} g_j u^{(j)}, \\
g \mid k, c &\sim N(0, (c(\Lambda_k + n^{-2} I)^q)^{-1}), \\
p(k) &\propto e^{-\gamma k}, \\
p(c) &\propto c^{a-1} e^{-bc}.
\end{aligned}
\tag{3.3}
$$

Our goal is to compute $f \mid D$ and use it to predict the unobserved labels.

3.2. Sampling scheme

We will use a reversible jump Markov chain Monte Carlo algorithm (Green (1995)) to sample from $f \mid D$ in the setup (3.3). This involves sampling repeatedly from the conditionals $p(z \mid c, k, g, D)$, $p(k, g \mid c, z, D)$, and $p(c \mid k, g, z, D)$. The joint move in k and g is the reversible jump step as k is the dimension of g. We detail these three steps in the following subsections.

3.2.1. Sampling from $p(z \mid c, k, g, D)$

As we identify $f = \sum_{j=1}^{k} g_j u^{(j)}$, we see that z has the same full conditional as in the setup in Hartog and Van Zanten (2018). Given c, f and D, the z_i's are independent and

$$
z_i \mid c, f, D \sim \begin{cases} N(f_i, 1), & \text{if } i \notin I^{\text{obs}}, \\ N_+(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 1, \\ N_-(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 0, \end{cases}
$$

where N_+ and N_- denote the normal distribution, conditioned to be positive or negative, respectively. Generating variables from these distribution can for example be done by a simple rejection algorithm or inversion (e.g. Devroye (1986), see Chopin (2011) for a more refined analysis).

3.2.2. Sampling from $p(k, g \mid c, z, D)$

Since, given z, we know all the y_i's and I^{obs} is independent of all other elements of the model, we have $p(k, g \mid c, z, D) = p(k, g \mid c, z)$. Due to the role of k in the model, we do not have conjugacy to draw from the exact conditional. Instead, we use a reversible jump step. To this end, we choose a proposal density $w(k', k)$. To generate a new draw for k, g we propose the following steps:

- draw a proposal $k' \sim w(\cdot, k)$;
- draw an independent uniform random variable v on $(0,1)$;
- if
$$v \leq \frac{p(z\,|\,c,k')p(k')w(k\,|\,k')}{p(z\,|\,c,k)p(k)w(k'\,|\,k)},$$
then accept the new proposal k' and for $j = 1, \dots k'$ draw
$$g_j \sim N\left(\frac{z^T u^{(j)}}{1 + c(\lambda_j + 1/n^2)^q}, \frac{1}{1 + c(\lambda_j + 1/n^2)^q}\right),$$
otherwise retain the old draws k and g.

We may choose a symmetric proposal distribution w, where, for example, the dimension can move a few steps up or down from the current level in a uniform, triangular or binomial way. This is similar to a random walk proposal. In that case, the ratio $w(k,k')/w(k',k) = 1$. We may integrate to see that

$$\begin{aligned} p(z\,|\,c,k) &= \int p(z\,|\,c,k,g)p(g\,|\,c,k)dg \\ &= (2\pi)^{-n/2}\left(\prod_{j=1}^{k} \frac{c(\lambda_j + 1/n^2)^q}{1 + c(\lambda_j + 1/n^2)^q}\right)^{1/2} e^{-\frac{1}{2}z^T z + \frac{1}{2}\sum_{j=1}^{k} \frac{(z^T u^{(j)})^2}{1 + c(\lambda_j + 1/n^2)^q}}, \end{aligned}$$

resulting in the following three cases:

$$\frac{p(z\,|\,c,k')}{p(z\,|\,c,k)} = \begin{cases} \left(\prod_{j=k'+1}^{k} \frac{1+c(\lambda_j+1/n^2)^q}{c(\lambda_j+1/n^2)^q}\right)^{1/2} e^{-\frac{1}{2}\sum_{j=k'+1}^{k} \frac{z^T u^{(j)}}{1+c(\lambda_j+1/n^2)^q}}, & \text{if } k' < k, \\ 1, & \text{if } k' = k, \\ \left(\prod_{j=k+1}^{k'} \frac{c(\lambda_j+1/n^2)^q}{1+c(\lambda_j+1/n^2)^q}\right)^{1/2} e^{\frac{1}{2}\sum_{j=k+1}^{k'} \frac{z^T u^{(j)}}{1+c(\lambda_j+1/n^2)^q}}, & \text{if } k' > k. \end{cases}$$

In our numerical experiments, we use $w(k',k) = k-2+B$, where $B \sim \text{Binom}(4, 0.5)$. In the following lemma we show detailed balance for this move, this implies that our proposed Markov chain has the correct stationary distribution (see e.g. Brooks et al. (2011)).

Lemma 3.2.1. *The above proposed steps satisfy the relation*
$$p(k,g\,|\,c,z)p((k,g) \to (k',g')) = p(k',g'\,|\,c,z)p((k',g') \to (k,g)),$$
where $p(A \to B)$ denotes the transition density from state A to state B.

Proof. The transition density from (k,g) to (k',g') is
$$p((k,g) \to (k',g')) = \min\left\{1, \frac{p(z\,|\,c,k')p(k')w(k,k')}{p(z\,|\,c,k)p(k)w(k',k)}\right\} w(k',k)p(g'\,|\,c,k',z).$$
Note that if the minimum is less than 1, the opposite move has a minimum larger than one. Using
$$p(k,g\,|\,c,z) = p(g\,|\,c,k,z)p(k\,|\,c,z),$$
and that the priors for c and k are independent, the assertion is verified. In case the minimum is greater than 1 can be dealt with in a similar way. □

3.2.3. SAMPLING FROM $p(c\,|\,k,g,z,D)$

We see that, given g, c is independent of the rest of the variables. In this case we have the usual normal-inverse gamma conjugacy giving

$$c\,|\,k,g \sim \Gamma\left(a+\frac{k}{2}, b+\frac{1}{2}\sum_{j=1}^{k}(\lambda_j+1/n^2)^q g_j^2\right).$$

3.2.4. OVERVIEW OF SAMPLING SCHEME

For convenience we summarize our sampling scheme.

Algorithm 3.1 Sampling scheme using truncation.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, initial values for $c$, $k$ and $g$.
Output: MCMC sample from the joint posterior $p(c,k,g,z\,|\,D)$.
 1: repeat
 2:     Compute $f \leftarrow \sum_{j=1}^{k} g_j u^{(j)}$ and for $i = 1,\dots,n$, draw independent
            $$z_i \sim \begin{cases} N(f_i,1), & \text{if } i \not\in I^{\text{obs}}, \\ N_+(f_i,1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 1, \\ N_-(f_i,1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 0. \end{cases}$$
 3:     Draw a proposal $k' \sim w(\cdot,k)$ and a uniform $v$ on $(0,1)$.
 4:
 5:     if
            $$v \le e^{-\gamma(k'-k)}\frac{w(k,k')}{w(k',k)}\begin{cases} \left(\prod_{j=k'+1}^{k}\frac{1+c(\lambda_j+1/n^2)^q}{c(\lambda_j+1/n^2)^q}\right)^{1/2} e^{-\frac{1}{2}\sum_{j=k'+1}^{k}\frac{z^T u^{(j)}}{1+c(\lambda_j+1/n^2)^q}}, & \text{if } k' < k, \\ 1, & \text{if } k' = k, \\ \left(\prod_{j=k+1}^{k'}\frac{c(\lambda_j+1/n^2)^q}{1+c(\lambda_j+1/n^2)^q}\right)^{1/2} e^{\frac{1}{2}\sum_{j=k+1}^{k'}\frac{z^T u^{(j)}}{1+c(\lambda_j+1/n^2)^q}}, & \text{if } k' > k. \end{cases}$$
        then
 6:         Set $k \leftarrow k'$ and for $j = 1,\dots k$ draw
            $$g_j \sim N\left(\frac{z^T u^{(j)}}{1+c(\lambda_j+1/n^2)^q}, \frac{1}{1+c(\lambda_j+1/n^2)^q}\right),$$
 7:     else
 8:         Retain $k$ and $g$.
 9:     end if
10:     Draw
            $$c \sim \Gamma\left(a+\frac{k}{2}, b+\frac{1}{2}\sum_{j=1}^{k}(\lambda_j+1/n^2)^q g_j^2\right).$$
11: until you have a large enough sample.
```

3.3. COMPUTATIONAL ASPECTS

If the underlying function f is smooth enough that we can approximate it with only a few $k \ll n$ eigenvectors, then the proposed algorithm needs an initial investment of $O(kn^2)$ to compute the first k eigenvalues and eigenvectors, in case these are not explicitly known for the graph under consideration. Step 6 in Algorithm 3.1 has complexity $O(kn)$ and is the most expensive step. In principle, it could be that $k = n$ and our method would be as complex as the algorithm proposed in Hartog and Van Zanten (2018). However, for very large graphs, it could be prohibitive to calculate the full eigendecomposition. One could compute a fixed number of eigenvalues and eigenvectors and if the Markov chain is about to step beyond this number one could either compute the next eigenvalue-eigenvector pair on the fly or reject the proposed k.

3.4. NUMERICAL RESULTS

In this section, we numerically assess scalability of the method and the sensitivity to the choice of the truncation level.

3.4.1. IMPACT OF THE TRUNCATION LEVEL

To assess the impact of the truncation level γ we first consider a basic example of simulated data on the path graph with $n = 500$ vertices. In this case, the eigenvalues of the Laplacian matrix are $\lambda_k = 4\sin^2(\pi(j-1)/(2n))$ with corresponding eigenvectors given by

$$u_i^{(j)} = \begin{cases} \frac{\sqrt{2}}{\sqrt{n}} \cos\left(\frac{\pi(i-\frac{1}{2})j}{n}\right), & j = 2, \dots, n, \\ \frac{1}{\sqrt{n}}, & j = 1, \end{cases} \tag{3.4}$$

for $i = 1, \dots, n$. We construct a function f_0 on the graph representing the ground truth by setting

$$f_0 = \sum_{j=1}^{n} a_j u^{(j)},$$

where we choose $a_j = \sqrt{n}(j-1)^{-1.5}\sin(j-1)$ for $j > 1$ and $a_1 = 0$. We simulate noisy labels Y_i on the graph vertices satisfying $P(Y_i = 1) = \ell_0(i) = \Phi(f_0(i))$, where Φ is the cdf of the standard normal distribution. Finally, we remove 20% of the labels at random to generate the set of observed labels Y^{obs}. Figure 3.1 shows the resulting soft label function ℓ_0 and the simulated noisy labels Y_i.

From the construction of our prior, we would like to spread out the mass in the prior on c and perhaps favor low values in the prior on k for computational efficiency. From the construction of the prior on k we see that high values for the parameter γ correspond to more prior mass on low values of k and low values spread out the prior mass over all possible values of k with limiting case $\gamma = 0$ corresponding to $p(k) \propto 1$. In Figure 3.2, we visualize the posterior for the soft label function ℓ for various values for γ. We have used the $a = b = 0$ and proposal probabilities $(0.0625, 0.25, 0.375, 0.25, 0.0625)$ for $k-2, \dots, k+2$, this corresponds to a Binom$(4, 0.5)$ proposal as mentioned in Section 3.2.2. The blue line is the posterior mean and the gray area depicts point-wise 95% credible intervals. The bottom plots are the posterior draws for k. We observe that a high γ results in low values

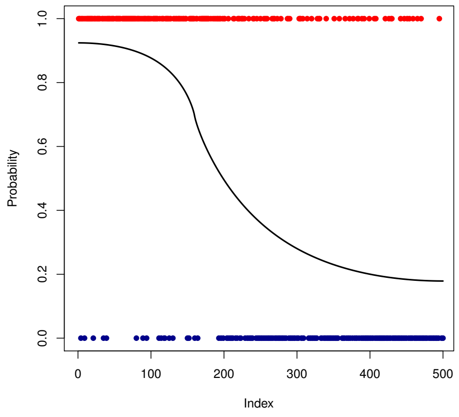


Figure 3.1: Soft label function $\ell_0 = \Phi(f_0)$ and simulated noisy label on a path graph with $n = 500$ nodes.

for k, as expected. However, if we choose γ too high, we might be oversmoothing as a result of taking too few eigenvectors. If we compare the cases $\gamma = 0$ and $\gamma = 0.1$ we observe only a little difference in the estimation performance, whereas the number of eigenvectors used in case of $\gamma = 0.1$ is only a fraction of the number of eigenvectors used in case of $\gamma = 0$.

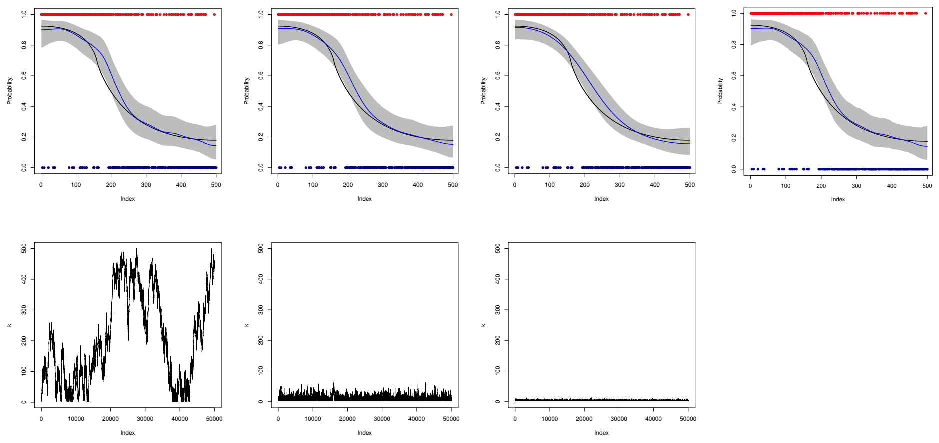


Figure 3.2: Top: Posteriors for the soft label function for $\gamma = 0, 0.1, 1$. The rightmost picture is the case $k = n$ for comparison. Bottom: The corresponding draws from the posterior of k. The black line is the true underlying soft label function. The blue line is the posterior mean and the gray area depicts point-wise 95% credible intervals. The red and blue dots are the observations.

We also consider a simulated example on a small-world graph obtained as a realization of the Watts-Strogatz model (Watts and Strogatz, 1998). The graph is obtained by first considering a ring graph of 1000 nodes. Then, we loop through the nodes and uniformly rewire each edge with probability 0.25. We keep the largest connected component and delete multiple edges and loops, resulting in a graph with 848 nodes as shown in Figure 3.3. We use the same construction of the observed data on the graph as in the previous example on the path graph. Our suggested rule-of-thumb of setting $\gamma = 20/n$ corresponds in the previous examples to $\gamma = 0.04$ and $\gamma = 0.024$, which in both cases ends up in using only a small fraction of the total number of eigenvectors, but it does not oversmooth too much. The results are in Figure 3.4.

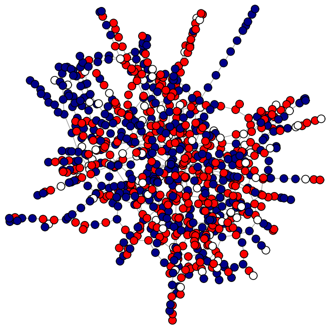


Figure 3.3: Small world graph with two types of labels. White nodes represent unobserved labels.

3.4.2. COMPUTATIONAL GAINS: MNIST DATA

The MNIST data set consists of images of handwritten digits. The images are size-normalized and centered. The dataset is publicly available at `http://yann.lecun.com/exbd/mnist`. We have selected the images of only the digits 4 and 9 from both the test set (1991 images) and the training set (11791 images). Our goal is to classify the images from the test set using the images from the training set. To turn this into a label prediction problem on a graph we construct a graph with $11791 + 1991 = 13782$ nodes representing the images. For each image we determine the 15 closest images in Euclidean distance between the projections on the first 50 principal components, similar to Bertozzi et al. (2018), Liang et al. (2007) and Belkin et al. (2006).

We use this example to explore the relative speedup of our proposed method with respect to the method proposed in Hartog and Van Zanten (2018) without truncation, where we also compare the difference in prediction accuracy. To this end, we take random subsamples of different sizes of the graph to illustrate what happens when the size of the graph grows. The ratio of 4's and 9's in the test and train set is kept constant and equal to that of the entire dataset. In Figure 3.5 we observe a dramatic difference in the computational time for the algorithm without truncation versus the algorithm with truncation, while the prediction performance is comparable.

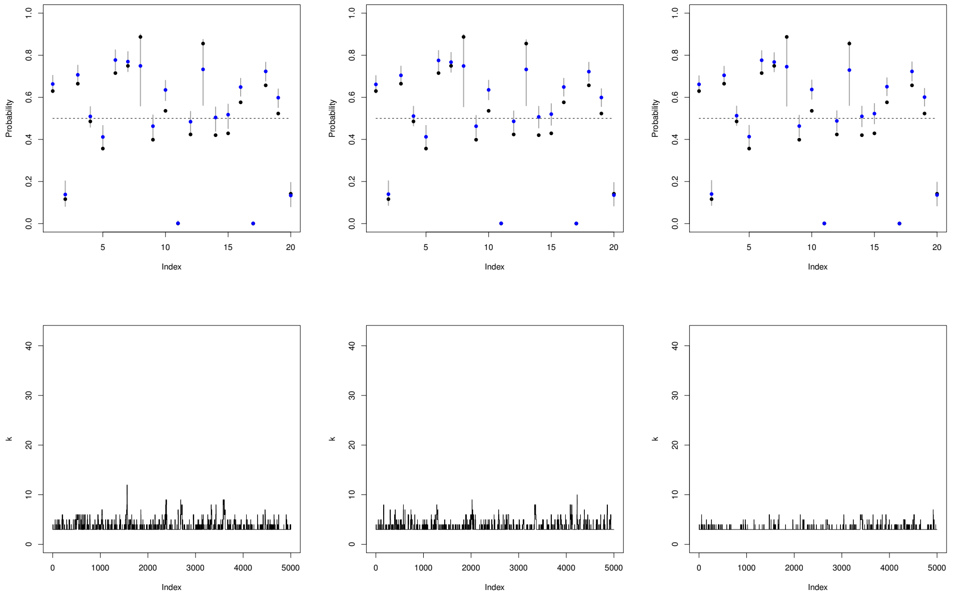


Figure 3.4: Top: Posteriors for the soft label function for $\gamma = 0, 0.1, 1$. Bottom: The corresponding draws from the posterior of k.

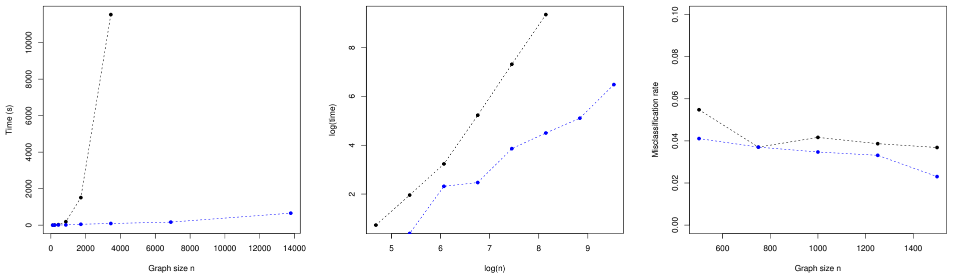


Figure 3.5: Left: Computation time versus graph size. Middle: Computation time versus graph size on a log-log scale. Right: The prediction performance on the different subgraphs. The blue line is our proposed truncation method, the black line is the method from Hartog and Van Zanten (2018). In both methods we set $a = b = 0$ and we choose $\gamma = 20/n$.

3.4.3. LARGE SCALE EXAMPLE: OBJECT TRACKING

To demonstrate the applicability of our proposed method in a large graph, where for example the untruncated method from Hartog and Van Zanten (2018) is prohibitive, we use a simulated object tracking application. As ground truth, we use the animation given by the following frames (Figure 3.6).

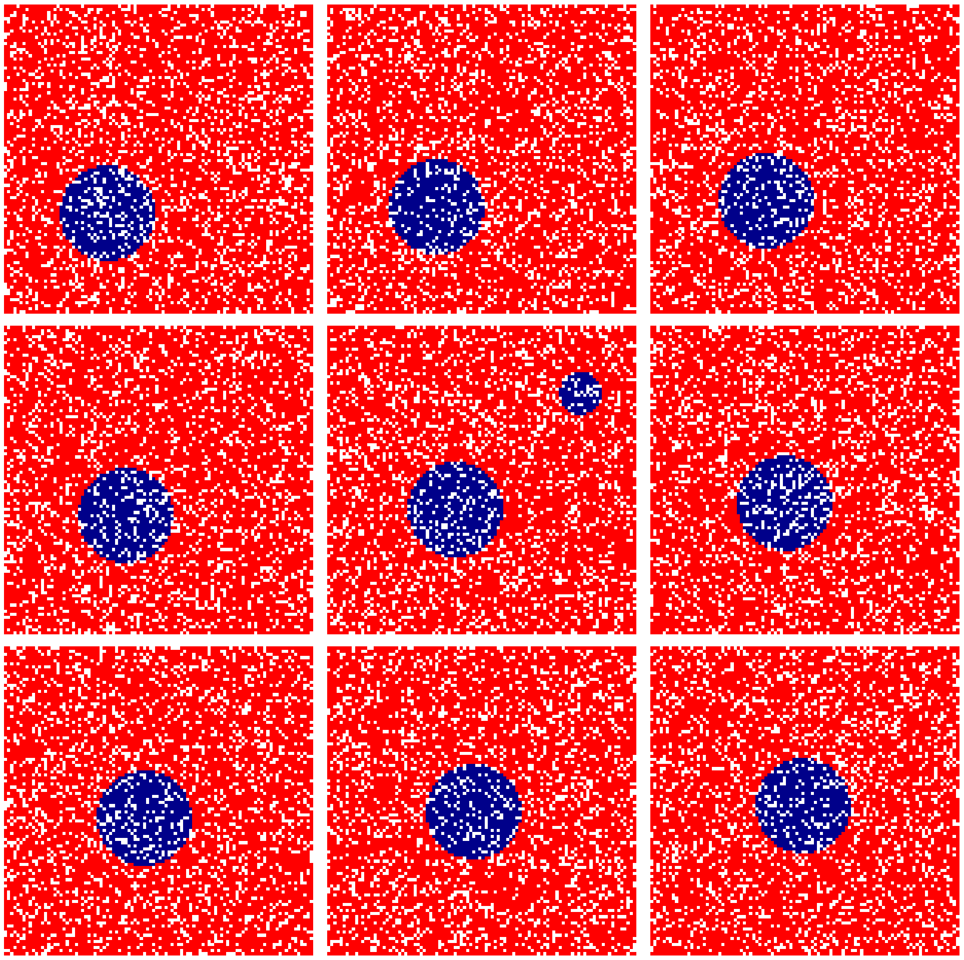


Figure 3.6: Moving object in a noisy environment, images are 100×100 pixels. The fifth image is corrupted with an additional object which should be removed.

The animation consists of 9 frames of 100×100 pixels. It represents a slowly moving blue ball on a red background. We removed the color of 20% of the pixels at random and added an additional ball in the fifth image to represent a corrupted frame. To convert the animation into a graph problem we connect neighboring pixels in each frame and with the corresponding pixels in the previous and next frame, resulting in a $100 \times 100 \times 9$ grid graph on a total of $n = 90000$ nodes as in Figure 3.7.

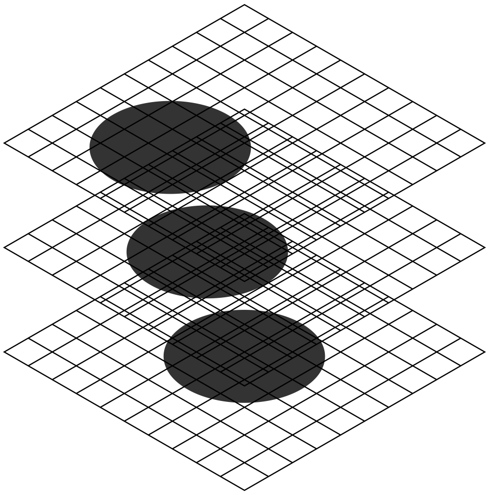


Figure 3.7: Schematic representation of the construction of the grid graph. Each pixel is also connected to the pixel at the same location in the frame before and after the current frame (these lines are omitted from the above representation for clarity).

We can explicitly compute the eigenvalues as $\lambda_i + \mu_j + \nu_k$ (cf. Theorem 3.5 in Mohar (1991)), where

$$\begin{aligned}\lambda_i &= 4\sin^2\left(\frac{\pi i}{200}\right), \quad i = 0,\dots,99,\\ \mu_j &= 4\sin^2\left(\frac{\pi j}{200}\right), \quad j = 0,\dots,99,\\ \nu_t &= 4\sin^2\left(\frac{\pi t}{18}\right), \quad t = 0,\dots,8.\end{aligned}$$

The corresponding eigenvectors are given by the tensor products $u^{(i)} \otimes v^{(j)} \otimes w^{(t)}$ of eigenvectors of the path graph with sizes 100 (u), 100 (v) and 9 (w) as in Equation (3.4). Using the noisy images, we estimate the location of the ball as shown in Figures 3.8 and 3.9,

In this particular example, we have an explicit interpretation of the dimensions of the graph. In this case, it would also make sense to make a difference between the space dimensions, corresponding to eigenvectors $u^{(i)}$ and $v^{(j)}$, and the time dimension, corresponding to eigenvectors $w^{(t)}$. Our framework allows for an extension, by using different priors for the truncation point in those two types of dimension. Instead of a single truncation in the model (3.3), we can use

$$f = \sum_{i,j}^{k} \sum_{t=1}^{l} g_{i,j,t} u^{(i)} \otimes v^{(j)} \otimes w^{(t)},$$

where $p(k) \propto e^{-\gamma k}$ and $p(l) \propto e^{-\delta l}$, are independent priors for the two different truncation levels. Alternatively, we can use the approach from Section 3.3 and compute a fixed number of eigenvalues and eigenvectors in each dimension and compute the next pair on the fly or reject the proposed move. We chose this last alternative in our example with 20 eigenvectors in each space dimension and all 9 in the time dimension.

We observe that the object is located in all images and that the additional object in frame 5 adds some noise in the probability estimates, but is ignored when we truncate at probability 0.5.

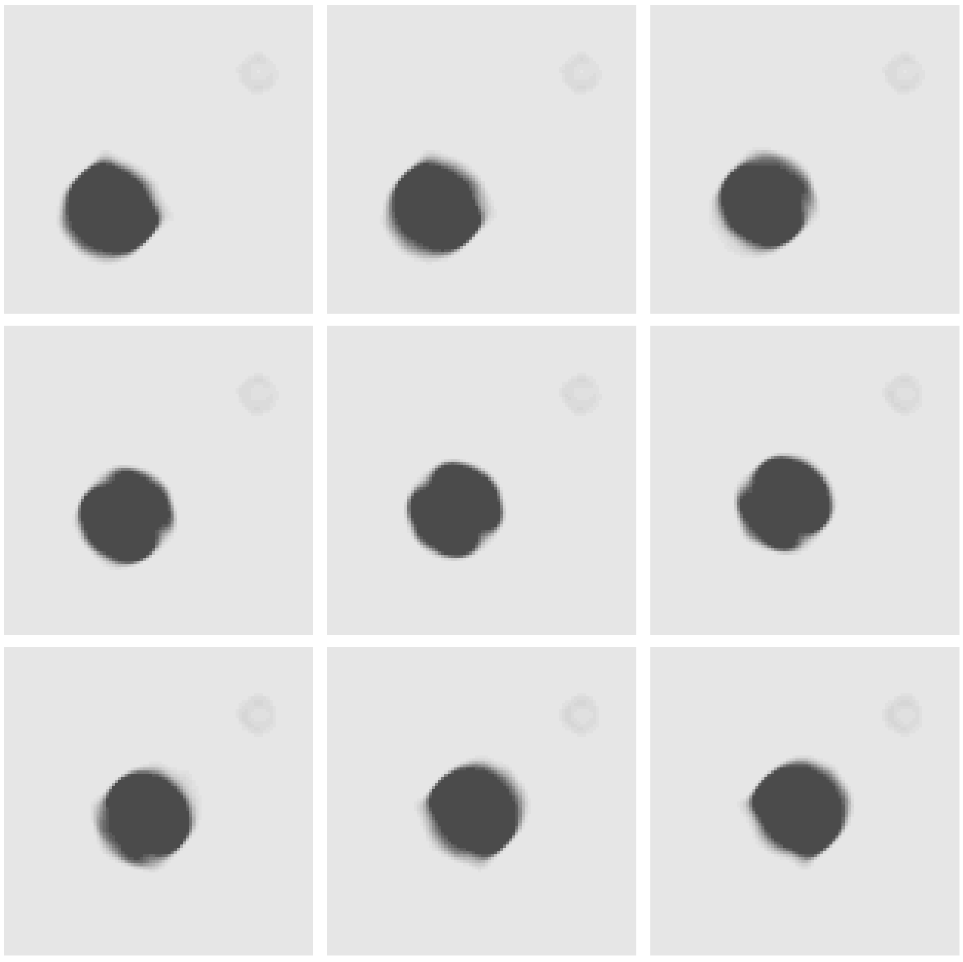


Figure 3.8: Estimated probabilities of location of object in a noisy environment. The gray scale represents a probability of being a black pixels where dark is close to 1 and light is close to 0.

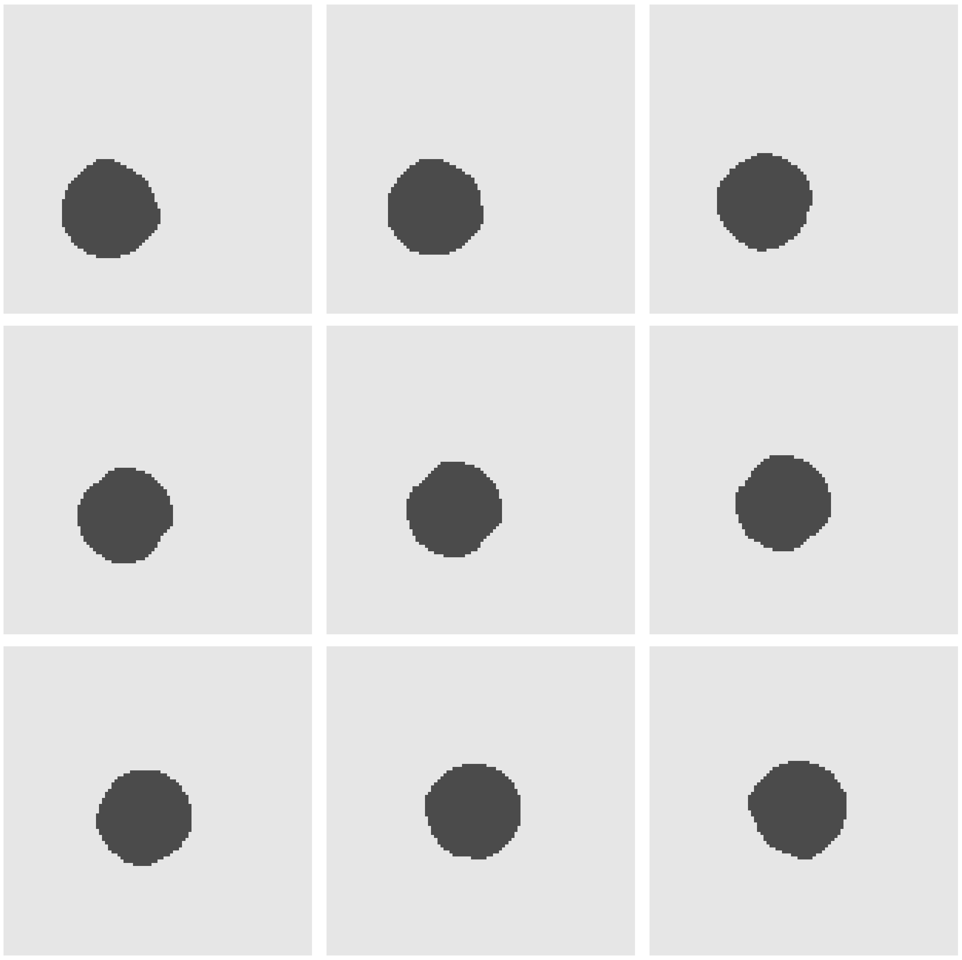


Figure 3.9: Estimated location of object in a noisy environment. We truncated the probabilities from Figure 3.8 at 0.5 to decide whether the pixels belong to the object or not.

An advantage of the Bayesian procedure that we use is that we obtain credible intervals as indicators of uncertainty in our prediction. The width of these intervals per pixel are shown in Figure 3.10. We observe that the uncertainty around the boundary of the object is relatively high, whereas it is relatively low inside and outside of the object.

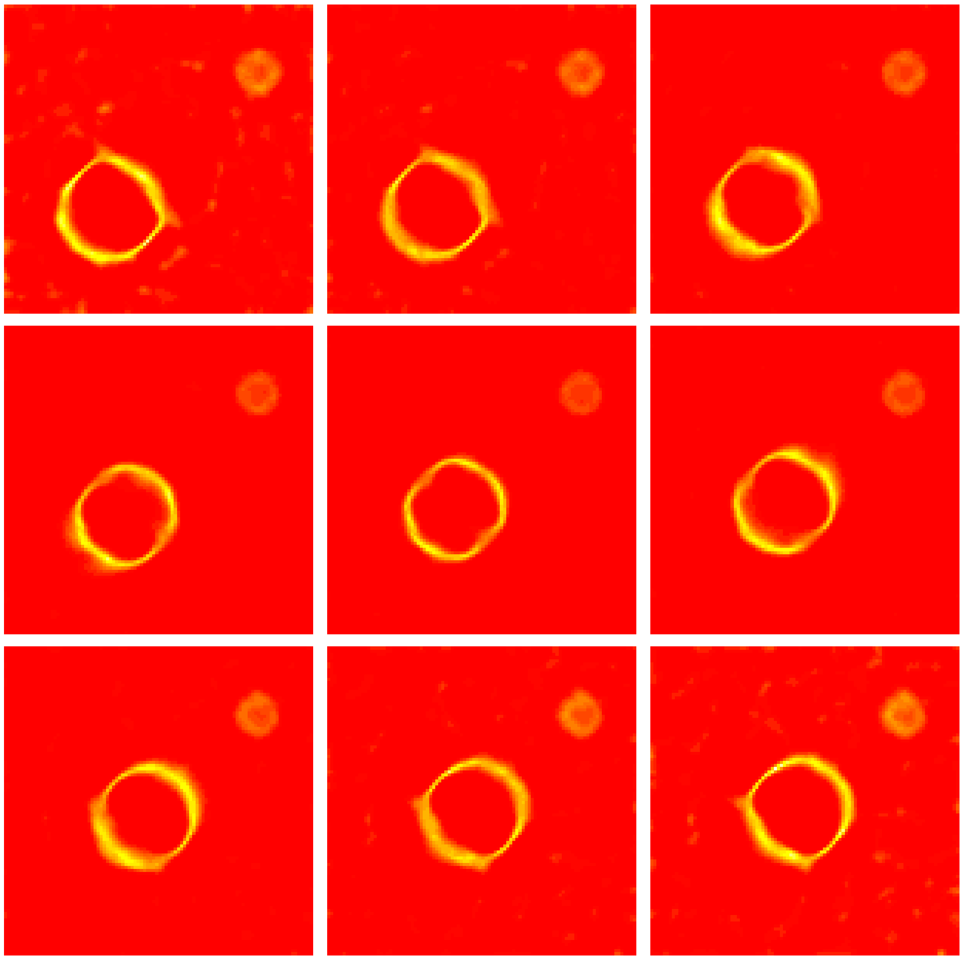


Figure 3.10: Uncertainty in of the predicted probability for each pixel computed as the width of the 95% credible interval. The heat map is from red (low uncertainty, width of credible interval close to 0) to white (high uncertainty, width of credible interval close to 0.25).

3.5. REGRESSION PROBLEM

In this section, we demonstrate the hierarchical Bayesian approach using truncation to solve the regression problem on a graph. We repeat the regression setting for completeness. The graph is denoted by $G = (V, E)$, where $V = \{1, \dots, n\}$. We assume that the noisy labels y are Gaussian with

$$y \mid f \sim N(f, \sigma^2 I),$$

where f is the function of interest. For simplicity, we assume σ^2 is fixed and known and that all noisy label are observed. We are interested in inferring f from the data y. Similar to the classification problem in the previous sections, we consider a Gaussian prior on f, which depends on the Laplacian matrix L, on a multiplicative scale parameter $c > 0$

and a truncation level k. The resulting prior is given by the truncated series

$$f = \sum_{j=1}^{k} g_j u^{(j)},$$

where

$$g \,|\, c, k \sim N(0, (c(\Lambda_k + n^{-2} I)^q)^{-1}),$$

for some fixed $q > 0$, where Λ_k is the restriction of the matrix of eigenvalues Λ, from the eigendecomposition $L = U\Lambda U^T$, to the first k rows and columns. The additional number n^{-2} is added to L to make it invertible as in Section 3.1. In this chapter, we have considered the ordinary gamma prior for the scaling parameter c,

$$c \sim \Gamma(a, b)$$

for $a, b > 0$. The improper case of $a = b = 0$ is also used in the literature (e.g. Choudhuri et al. (2007); Liang et al. (2007)). In Section 3.4, we have seen that this is a reasonable choice in our setting as well. For the truncation level, we use the exponential prior

$$p(k) \propto e^{-\gamma k},$$

for $k = 1, \dots, n$. Motivated by the numerical experiments in Section 3.4, we use the rule-of-thumb $\gamma = 20/n$. The four distributions $y \,|\, k, g$, $g \,|\, c, k$, c and k form the hierarchical model.

To sample from the posterior $f \,|\, y$, we use a Gibbs sampler, as in Algorithm 1.1, combined with a reversible jump step, as in Algorithm 1.3. We can naturally partition the unknown parameter as (c, g, k). To sample from the joint posterior $c, g, k \,|\, y$ we iteratively sample from the conditionals $k, g \,|\, c, y$, using a reversible jump move, and $c \,|\, k, g, y$.

To apply Algorithm 1.3 for a reversible jump in $g, k \,|\, c, y$, we need to compute the densities $p(k \,|\, c, y)$ and $p(g \,|\, k, c, y)$. We write U_k for the restriction of U to the first k columns and find

$$g \,|\, c, k, y \sim N(\sigma^{-2}(\sigma^{-2} I + c(\Lambda_k + n^{-2} I)^q)^{-1} U_k^T y, (\sigma^{-2} I + c(\Lambda_k + n^{-2} I)^q)^{-1})$$

and

$$p(k \,|\, c, y) \propto p(k)(\sigma^{-2})^{\frac{n}{2}} \left(\prod_{j=1}^{k} \frac{c(\lambda_j + n^{-2})^q}{\sigma^{-2} + c(\lambda_j + n^{-2})^q} \right)^{\frac{1}{2}} e^{-\frac{1}{2}\frac{y^T y}{\sigma^2} + \frac{1}{2}\sum_{j=1}^{k} \frac{(\sigma^{-2} y^T u^{(j)})^2}{\sigma^{-2} + c(\lambda_j + n^{-2})^q}}.$$

The conditional $p(c \,|\, k, g, y)$ can be found using the normal-inverse gamma partial conjugacy. It should be noted that given (k, g), y is independent of c, so $c \,|\, k, g, y$ does not depend on y. We find

$$c \,|\, k, g, y \sim \Gamma\left(a + \frac{k}{2}, b + \frac{1}{2} \sum_{j=1}^{k} (\sigma^{-2} + (\lambda_j + n^{-2})^q) g_j^2 \right).$$

We see that the improper prior with $a = b = 0$ results in a proper posterior.

These steps can be summarized in the following algorithm:

Algorithm 3.2 Simple sampling scheme for the regression problem using truncation.

```
Input: Data $D = \{(i, y_i) : i = 1, \dots, n\}$, variance $\sigma^2$, initial values $c$, $k$ and $g$.
Output: MCMC sample from the joint posterior $c, k, g \mid y$.
1: repeat
2:     Draw a proposal $k' \sim w(\cdot, k)$ and uniform $v$ on $(0,1)$.
3:     if
           $$v \leq \frac{p(k)}{p(k')} \frac{w(k,k')}{w(k',k)} \begin{cases} \left(\prod_{j=k'+1}^{k} \frac{\sigma^{-2}+c(\lambda_j+n^{-2})^q}{c(\lambda_j+n^{-2})^q}\right)^{\frac{1}{2}} e^{-\frac{1}{2}\sum_{j=k'+1}^{k} \frac{(\sigma^{-2}y^T u^{(j)})^2}{\sigma^{-2}+c(\lambda_j+n^{-2})^q}}, & k' < k \\ 1, & k' = k \\ \left(\prod_{j=k+1}^{k'} \frac{c(\lambda_j+n^{-2})^q}{\sigma^{-2}+c(\lambda_j+n^{-2})^q}\right)^{\frac{1}{2}} e^{\frac{1}{2}\sum_{j=k+1}^{k'} \frac{(\sigma^{-2}y^T u^{(j)})^2}{\sigma^{-2}+c(\lambda_j+n^{-2})^q}}, & k' > k \end{cases}$$
       then
4:         Update $k \leftarrow k'$ and for $j = 1, \dots, k$, draw
               $$g_j \sim N(\sigma^{-2}(\sigma^{-2} + c(\lambda_j + n^{-2})^q)^{-1} y^T u^{(j)}, (\sigma^{-2} + c(\lambda_j + n^{-2})^q)^{-1}).$$
5:     else
6:         Retain $g$ and $k$.
7:     end if
8:     Draw
               $$c \sim \Gamma\left(a + \frac{k}{2}, b + \frac{1}{2}\sum_{j=1}^{k} (\sigma^{-2} + (\lambda_j + n^{-2})^q) g_j^2\right).$$
9: until you have a large enough sample.
```

As a proposal, we use $k' = k - K/2 + B$, where $B \sim \text{Binom}(K, 1/2)$ for $K = 4$. When σ^2 is unknown, we can put an independent prior $p(\sigma^2)$ on σ^2, or rather, an independent prior $p(\sigma^{-2})$ on the precision σ^{-2}. We include a draw from $\sigma^{-2} \mid c, k, g, D$ in Algorithm 3.2 to sample from the joint posterior $g, k, c, \sigma^{-2} \mid D$. If we take a gamma prior for σ^{-2} with shape s and rate t, we see that

$$\sigma^{-2} \mid c, k, g, D \sim \Gamma\left(s + \frac{n}{2}, t + \frac{1}{2}\sum_{i=1}^{n} y_i^2 + \frac{1}{2}\sum_{j=1}^{k} (g_j^2 - 2 g_j y^T u^{(j)})\right).$$

To account for missing data, we can include the missing observations y^{miss} as latent variables and impute them prior to line 2 of Algorithm 3.2 in every iteration using

$$y^{\text{miss}} \mid c, k, g, \sigma^{-2}, D \sim N(U_k^{\text{miss}} g, (1/\sigma^{-2}) I),$$

where U_k^{miss} is the restriction of U_k to the rows corresponding to the missing observations. The other steps in Algorithm 3.2 remain unchanged as we simply use $y = (y^{\text{obs}}, y^{\text{miss}})$. A sampling scheme for the regression problem with missing data and unknown noise level is Algorithm 3.3.

Algorithm 3.3 Sampling scheme for the regression problem using truncation.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, initial values $c$, $k$, $g$ and $\sigma^{-2}$.
Output: MCMC sample from the joint posterior $y^{\text{miss}}, c, k, g, \sigma^{-2} \mid D$.
1: repeat
2:     For $i \notin I^{\text{obs}}$, draw
           $$y_i \sim N(U_{i,1\cdots k}g, 1/\sigma^{-2}).$$
3:     Draw a proposal $k' \sim w(\cdot, k)$ and uniform $v$ on $(0,1)$.
4:     if
           $$v \le \frac{p(k)}{p(k')}\frac{w(k,k')}{w(k',k)} \begin{cases} \left(\prod_{j=k'+1}^{k} \frac{\sigma^{-2}+c(\lambda_j+n^{-2})^q}{c(\lambda_j+n^{-2})^q}\right)^{\frac{1}{2}} e^{-\frac{1}{2}\sum_{j=k'+1}^{k}\frac{(\sigma^{-2}y^T u^{(j)})^2}{\sigma^{-2}+c(\lambda_j+n^{-2})^q}}, & k' < k \\ 1, & k' = k \\ \left(\prod_{j=k'+1}^{k} \frac{c(\lambda_j+n^{-2})^q}{\sigma^{-2}+c(\lambda_j+n^{-2})^q}\right)^{\frac{1}{2}} e^{\frac{1}{2}\sum_{j=k'+1}^{k}\frac{(\sigma^{-2}y^T u^{(j)})^2}{\sigma^{-2}+c(\lambda_j+n^{-2})^q}}, & k' > k \end{cases}$$
        then
5:         Update $k \leftarrow k'$ and for $j = 1, \ldots, k$, draw
           $$g_j \sim N(\sigma^{-2}(\sigma^{-2}+c(\lambda_j+n^{-2})^q)^{-1} y^T u^{(j)}, (\sigma^{-2}+c(\lambda_j+n^{-2})^q)^{-1}).$$
6:     else
7:         Retain $g$ and $k$.
8:     end if
9:     Draw
           $$c \sim \Gamma\left(a + \frac{k}{2}, b + \frac{1}{2}\sum_{j=1}^{k}(\sigma^{-2} + (\lambda_j + n^{-2})^q)g_j^2\right).$$
10:    Draw
           $$\sigma^{-2} \sim \Gamma\left(s + \frac{n}{2}, t + \frac{1}{2}\sum_{i=1}^{n} y_i^2 + \frac{1}{2}\sum_{j=1}^{k}(g_j^2 - 2g_j y^T u^{(j)})\right).$$
11: until you have a large enough sample.
```

3.5.1. TRAFFIC FLOW ESTIMATION

To illustrate the regression problem, we consider a simulated example of traffic flow estimation inspired by Aldous and Shun (2010) and Bickel et al. (2007). Traffic data is collected by authorities to measure the performance of the road network in terms of flow, occupancy and speed at locations in the network. A common type of sensor used for traffic flow estimation is a *loop detector* that counts the number of passing vehicles over a certain time interval. However, the resulting loop data is often missing or invalid due to communication errors or malfunction. Missing and bad observations are imputed from the neighboring sensor location using a simple average or a regression approach.

We simulate a road network between cities based on a toy model. We sample 500 points uniformly on the unit square and compute the *relative neighborhood graph* by connecting nodes u and v if there does not exist a node w such that

$$\max\{\|u-w\|, \|v-w\|\} < \|u-v\|.$$

In Figure 3.11 this definition is illustrated, nodes u and v are connected if there is no node in the gray area. The resulting graph is simple and connected. Relative neighborhood graphs and related proximity graphs, such as the nearest neighbor graph in Section 3.4.2, can be used as a toy models for road networks (see Aldous and Shun (2010)).

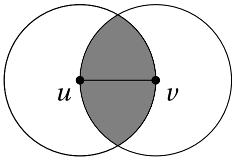


Figure 3.11: Construction of a relative neighborhood graph. Nodes u and v are connected if there is no point closer to both u and v as they are to each other, corresponding to the gray area not containing a node.

The resulting graph is the left graph in Figure 3.12. As traffic flow is a property of the roads in the city graph, we construct its *line graph*, by associating a vertex with each road in the city graph and connecting two vertices if the corresponding roads in the city graph have a city in common. This is the right graph in Figure 3.12 and is the road graph G we work with. It has $n = 617$ vertices.

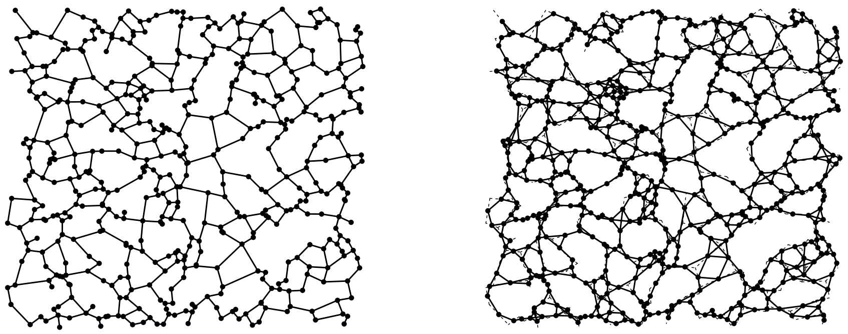


Figure 3.12: Left: A road network between cities as a realization of the relative neighborhood graph on 500 random uniform points on the unit square. Right: The corresponding line graph, where the nodes correspond to the roads of the left graph. The dashed line is the original graph.

We construct a true traffic flow function f_0 on the graph by setting

$$f_0 = \sum_{j=1}^{n} a_j u^{(j)},$$

where $u^{(j)}$ are the eigenvectors of the Laplacian matrix $L = U\Lambda U^T$ of G. To construct the coefficients a_j we consider the path graph with vertices $x_i = i/n$ for $i = 1,\ldots,n$ and the function $5+f(x_i)$, where f is given by Equation (1.8). We compute a_j as the coefficients of $5+f$ on the path graph and use them in our graph G, generalizing the function on the path graph to our graph G. The resulting underlying traffic flow is depicted in Figure 3.13 on both the city graph and the road graph G.

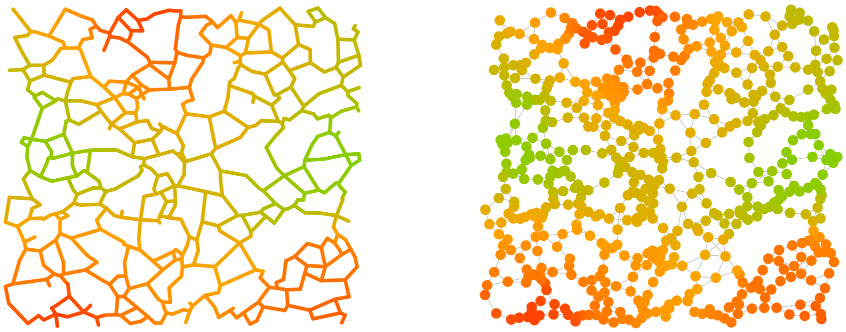


Figure 3.13: Left: The corresponding traffic flow on the city graph, where the flow is observed on the edges. Right: The traffic flow on the road graph G. Traffic intensity ranges from green (light traffic) to red (heavy traffic) and is observed on the nodes.

To construct a noisy observation of the traffic flow, we add independent noise $\epsilon_i \sim N(0,1)$ to each node $i = 1, \dots, n$ and remove 123 of our observations as depicted in Figure 3.14.

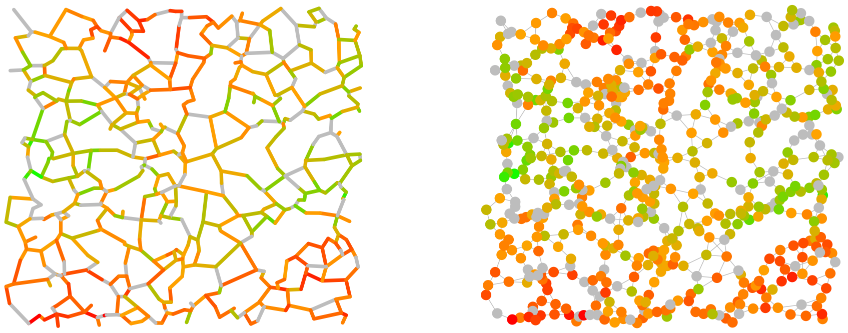


Figure 3.14: The noisy observation of the underlying traffic flow in Figure 2.24. The missing observations are in gray.

We estimate the traffic flow using Algorithm 3.3 with hyperparameters $a = b = 0$, $s = t = 1$ and $q = 1$. We numerically determine $r = 1.67$ from the spectrum of the Laplacian matrix as shown in Figure 3.15. We choose prior smoothness α not too large. The true function in Equation (1.8) is very smooth, but we will see that the truncated series prior is flexible enough to truncate early for posterior smoothness. The resulting posterior mean for $f = Ug$ is shown in Figure 3.16. To illustrate the uncertainty in our estimate, we also plot the estimated traffic flow along a path from the north-west corner of the map to the south-east corner in Figure 3.17 together with point-wise 95% credible intervals. An average of 7.8 terms are used to construct the estimate.

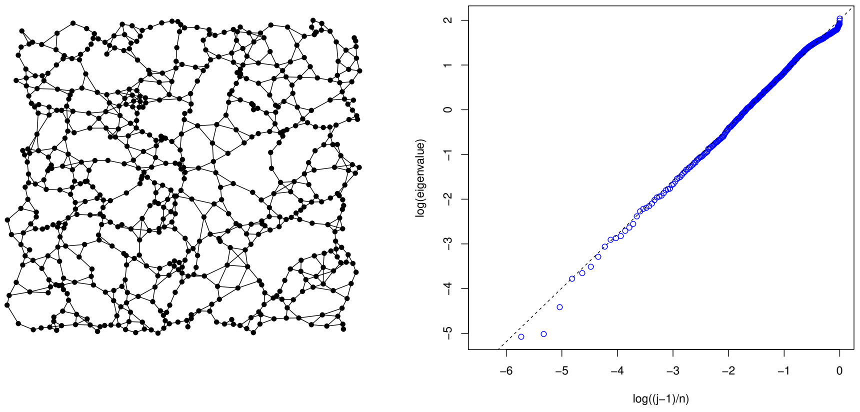


Figure 3.15: Left: The road graph. Right: The eigenvalues of the Laplacian matrix of the road graph. The dashed line corresponds to geometry number $r = 1.67$ based on a linear fit with slope 1.24. We have used the first 35% of the eigenvalues with exception of the first three.

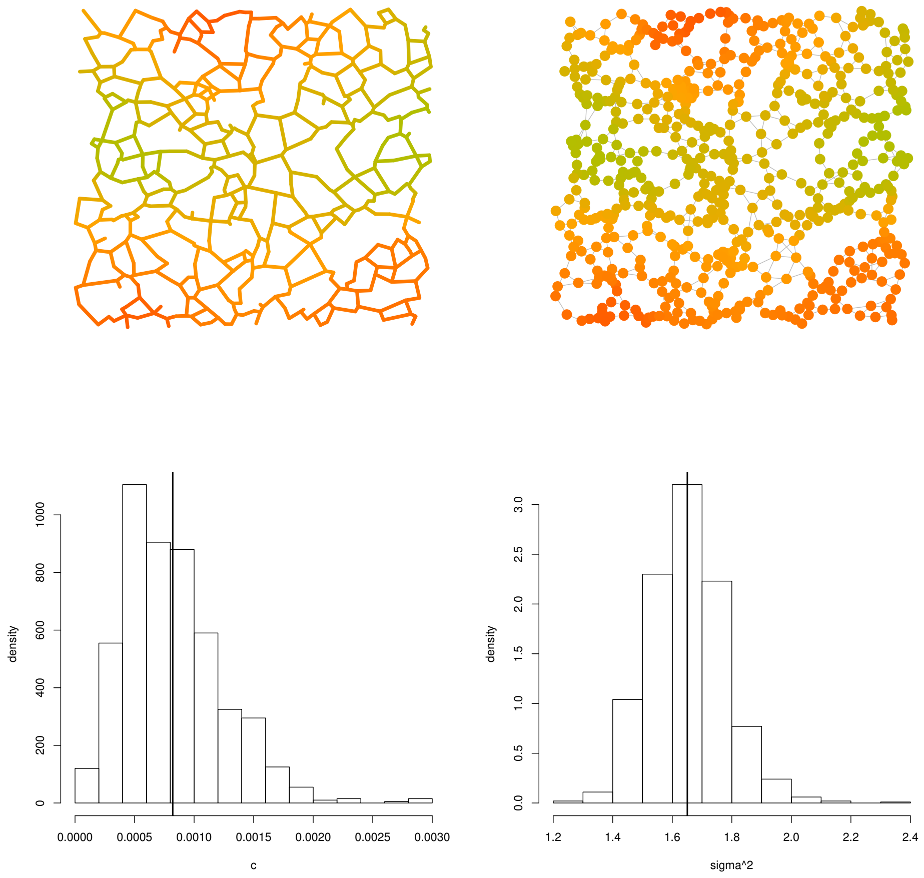


Figure 3.16: The posterior mean for the traffic flow data from Figure 3.14. Bottom: Histograms for the posterior distribution of c and σ^2. The vertical black lines are the corresponding posterior means.

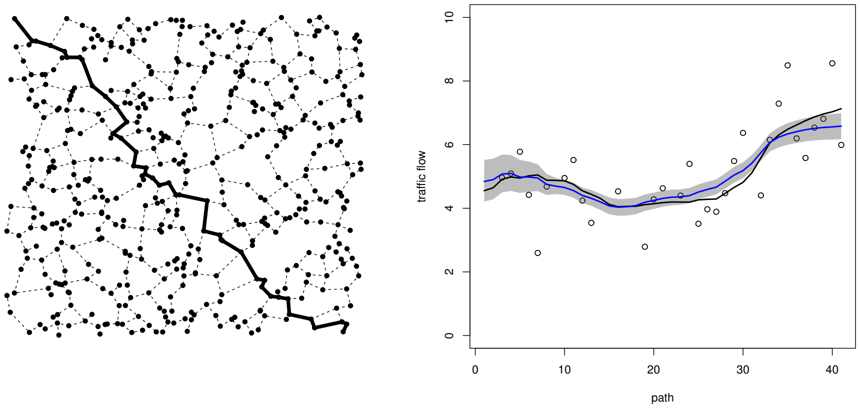


Figure 3.17: Left: A path in the city graph. Right: Posterior mean (blue) and point-wise 95% credible interval (gray area) for the traffic flow along the path. The points are the noisy observations and the black line is the true underlying traffic flow function.

3.6. CONCLUDING REMARKS

We have described an implementation of a nonparametric Bayesian approach to solve regression and binary classification problems on graphs. We have considered a hierarchical Bayesian approach with a randomly scaled Gaussian series prior as in Hartog and Van Zanten (2018), but with a random truncation point. We have implemented the procedure using a reversible jump Markov chain Monte Carlo algorithm.

Our numerical experiments suggest that good results for classification are obtained using Algorithm 3.1 using hyperparameters $a = b = 0$ and $\gamma = 20/n$. We find that in the examples we studied, the random truncation point results in a superior performance compared to the method proposed in Hartog and Van Zanten (2018) in terms of computational effort, while the prediction performance remains comparable. We have also demonstrated that our proposed method is scalable to large graphs and, with some adjustment, is applicable to the regression problem.

4 EMPIRICAL BAYES

In this chapter, we describe an implementation of a nonparametric Bayesian approach to solve regression and binary classification problems on graphs. We consider a empirical Bayesian approach with a prior that is constructed by truncating a series expansion of the soft label function using the graph Laplacian eigenfunctions as basis functions. The series is multiplied by a scale parameter for more flexibility. The scale parameter and truncation level are treated as unknown hyperparameters and are chosen such that the (approximate) marginal likelihood is maximized. To compute the marginal likelihood in the classification problem, we use a Laplace approximation. We demonstrate our method using simulated data examples.

In this chapter, we consider regression and classification problems on graphs. Our setup is the same as in Chapter 2 and Chapter 3 (Hartog and Van Zanten (2018)), but included here for completeness. Applications of regression and classification problems on graphs arise in, for example the prediction of biological function of a protein in a protein-protein interaction graph (e.g. Kolaczyk (2009); Nariai et al. (2007); Sharan et al. (2007)) or in graph-based semi-supervised learning (e.g. Belkin et al. (2004); Sindhwani et al. (2007)). We have problems in mind in which the graph is given by the application context. In the classification problem, the vertices of the graph can have (two) different values, corresponding the the possible classes of the vertices. In the regression problem, the label have real-valued labels. The available data in both cases is a noisy observation of some of the labels. We aim to classify the unobserved labels correctly.

The main idea behind our approach is that the information about the labels of neighboring vertices should typically have predictive power for the vertex label of interest. In our setting, we say that there exists a soft label function that determines the observed labels. This soft label function should vary smoothly over the graph. Moreover, we approximate the soft label function with a function of lower dimension.

We endow the soft label function with a prior distribution and determine the corresponding posterior. This posterior distribution can be used for prediction. As in Hartog and Van Zanten (2019), there are tuning parameters to be set. As with many nonparametric methods, the correct choice of bandwidth, smoothness or regularization parameters can be hard to determine. Our case is no exception, however in Hartog and Van Zanten (2019) a hierarchical Bayesian approach is presented, where the tuning parameters are also endowed with a prior distribution and the data automatically determines the appropriate values for it. A drawback of the proposed implementation is that sampling from the posterior distribution is time consuming, because it is done via a Markov chain Monte Carlo algorithm. In the present chapter, we propose an empirical Bayes approach, where we predict the unobserved labels in two steps. We use the scaled and truncated Gaussian series prior from Hartog and Van Zanten (2019) as a prior for the soft label function. First, we determine the optimal value for the tuning parameters, the regularization parameter and the truncation level, in terms of maximal marginal likelihood. As the marginal likelihood is intractable, we approximate it using Laplace approximation and maximize the approximated likelihood instead. Second, we fix the tuning parameters at the values attaining the maximal approximated likelihood and sample from the posterior. As a faster alternative, we can sample from the Laplace approximation to the posterior. In this two-step estimation approach, most of the computational costs is the first step. Note that the observed data is used in both steps.

The rest of this chapter is organized as follows. In the next section, we describe the problem setting and the priors we consider. Section 4.2 describes the Laplace approximation we use for the posterior distribution. Section 4.3 describes how the tuning parameters are selected using the data and Section 4.4 discusses how to use the fixed tuning parameters to sample from the posterior distribution. We also discuss the computational aspects of these algorithms. Section 4.5 consists of several examples to illustrate the different steps in our proposed algorithm, its strengths and possible weaknesses. In Section 4.6, we apply our empirical Bayes approach to the regression problem. Concluding remarks are given in Section 4.7.

4.1. OBSERVATION MODEL AND PRIOR

4.1.1. OBSERVATION MODEL

The observation model is similar to Hartog and Van Zanten (2018) and Hartog and Van Zanten (2019). We start with a connected, simple undirected graph $G = (V, E)$, with $\#V = n$ vertices denoted by $V = \{1, 2, \dots, n\}$. Associated to every vertex i is a noisy hard label y_i. We assume the y_i's are independent Bernoulli variables, with

$$P(y_i = 1) = 1 - P(y_i = -1) = \ell(i),$$

where $\ell : V \to (0, 1)$ is an unobserved function on the vertices of the graph, the so-called soft label function. Note that we use $\{-1, 1\}$ instead of the usual $\{0, 1\}$, for notational reasons later on. We observe only a subset $Y^{\text{obs}} \subset \{y_1, \dots, y_n\}$ of all the noisy labels. This can be a random subset of all the $\{y_1, \dots, y_n\}$, generated in an arbitrary way, but independent of the values of the labels. Note that in this setup we either observe the label of a vertex or not, so multiple observations of the same vertex are not possible.

4.1.2. PRIOR ON THE SOFT LABEL FUNCTION

The prior on the soft label function is the same as in Hartog and Van Zanten (2019). We repeat its construction here for completeness. Our prediction method consists in first inferring the soft label function ℓ from Y^{obs} and subsequently predicting the hard labels by thresholding. We take a Bayesian approach which is nonparametric, in the sense that we do not assume that ℓ belongs to some low-dimensional, for instance generalized linear family of functions.

To put a prior on ℓ we use a link h, for example, in case of probit classification $h = \Phi$ (i.e. the cdf of the standard normal distribution), to write $\ell = h(f)$ for some function $f : V \to \mathbb{R}$ and then put a prior on f. We consider a Gaussian series prior on f where each term is the series is an eigenvector of the Laplacian matrix L of the graph. Recall that this is the matrix defined as $L = D - A$, where D is the diagonal matrix of vertex degrees and A is the adjacency matrix of the graph. If we write the eigendecomposition $L = U\Lambda U^T$, then we can write any function f as

$$f = Ug = \sum_{j=1}^{n} g_j u^{(j)},$$

where $u^{(j)}$ is the j-th eigenvector and g_j its coefficient. The ordering of the eigenvectors in the above series is assumed to be according to the sizes of the eigenvalues, so $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_n$. To construct a prior on f we truncate the above series after k terms and employ an independent Gaussian prior on the coefficients g_j with mean zero and precision (inverse covariance) $e^c \lambda_j^q$, where c is a scale parameter, λ_j the eigenvalue of L corresponding to the jth eigenvector and $q > 0$ a fixed power. The lowest eigenvalue of the Laplacian matrix is 0 however, so it is not invertible. Therefore, one might add a small number ϵ to all eigenvalues, corresponding to eigendecomposition $U(\Lambda + \epsilon)U^T$ or set λ_1 equal to the smallest positive eigenvalue λ_2. In the following we assume either option is chosen and just write λ for the adjusted eigenvalues. We then have

$$\begin{aligned} \ell &= h(f), \\ f &= \sum_{j=1}^{k} g_j u^{(j)}, \\ g \mid c, k &\sim N(0, e^{-c}\Lambda^{-q}). \end{aligned}$$

We might also choose the logit link function given by $h(x) = 1/(1+e^{-x})$ instead of the probit link function. It slightly changes the model, but might have better numerical properties as explained in Section 4.2.1. We will estimate f in a two step procedure, namely by first estimating the hyperparameters k and c from the data, and then estimate f.

4.1.3. MISSING LABELS

We consider the situation in which we do not observe all the labels y_i, but only a certain subset Y^{obs}. The labels that we observe, are observed only once. The precise mechanism that determines which y_i's we observe and which ones are missing is not important for the algorithm we propose. We only assume that it is independent of the other elements of the model. Specifically, we assume that for some arbitrary distribution μ on the collection 2^V of subsets of the vertices, a set of vertices I^{obs} is drawn and that we see which vertices are selected and what the corresponding noisy labels are. In other words, the observed data is $D = \{(i, y_i) : i \in I^{\text{obs}}\}$.

All in all, the full model we will work with is the following:

$$\begin{aligned} D &= \{(i, y_i) : i \in I^{\text{obs}}\}, \\ I^{\text{obs}} &\sim \mu, \\ y_i \mid f_i &\sim \text{independent Bernoulli}(h(f_i)), \quad i \in I^{\text{obs}}, \\ f &= Ug, \\ g \mid c, k &\sim N(0, e^{-c}\Lambda^{-q}), \end{aligned}$$

where the matrices U and Λ are the appropriate-sized submatrices of our eigendecomposition $L = U\Lambda U^T$, i.e. U is the $|I^{\text{obs}}| \times k$ matrix with rows corresponding to the observed labels and the first k rows and Λ is the $k \times k$ matrix corresponding to the first k rows and columns. Our goal is to first estimate the optimal hyperparameters c^* and k^* using the data D, and thereafter fixing them to infer $g \mid D, c^*, k^*$ which, in turn, we can use to predict the unobserved labels. The unobserved hard labels can be inferred by thresholding, i.e. $\text{sign}(Ug)$.

4.2. LAPLACE APPROXIMATION

We select the hyperparameters c and k by maximizing the marginal likelihood $p(Y^{\text{obs}} \mid c, k)$. Our approach is similar to the kernel-based approach in Rasmussen and Williams (2006). We can write the likelihood as

$$p(Y \mid g, c, k) = \prod_{i \in I^{\text{obs}}} p(Y_i \mid c, k, g),$$

where

$$p(Y_i \mid c, k, g) = h(Y_i (Ug)_i)$$

and

$$(Ug)_i = \sum_{j=1}^{k} g_j u_i^{(j)}.$$

Note that we chose that the two labels are $Y_i \in \{-1, 1\}$ so that the above expression is simple. The prior on g has density

$$p(g \mid c, k) = (2\pi)^{-k/2} e^{ck/2} \left(\prod_{j=1}^{k} \lambda_j^{q/2} \right) e^{-\frac{1}{2} e^c g^T \Lambda^q g}$$

and so the joint distribution is

$$p(Y, g \mid c, k) = p(Y \mid g, c, k) p(g \mid c, k).$$

The marginal likelihood is computed by integrating out g:

$$p(Y \mid c, k) = \int p(Y, g \mid c, k) dg.$$

To efficiently compute this integral, we employ Laplace approximation. We write the integral as

$$p(Y \mid c,k) = \int e^{\Psi(g)}\, dg,$$

where

$$\begin{aligned}\Psi(g) &= \log p(Y \mid g,c,k) + \log p(g \mid c,k)\\ &= \sum_{i\in I^{\mathrm{obs}}} h(Y_i(Ug)_i) - \frac{k}{2}\log 2\pi + \frac{ck}{2} + \frac{q}{2}\sum_{j=1}^{k}\log\lambda_j - \frac{e^c}{2}\sum_{j=1}^{k} g_j^2\lambda_j^q.\end{aligned}$$

The Laplace approximation consists of approximating Ψ with its second order Taylor polynomial in the location of the maximum $\hat{g}$, so

$$\Psi(g) \approx \Psi(\hat{g}) - \frac{1}{2}(g-\hat{g})^T A(g-\hat{g}),$$

where $A = -H\Psi(\hat{g})$ is the negative Hessian matrix of Ψ in $\hat{g}$. We will see that A is positive definite and so $\Psi(g)$ is concave and therefore has a unique maximum $\hat{g}$. The approximated marginal likelihood is given by

$$q(c,k) = \int e^{\Psi(\hat{g})-\frac{1}{2}(g-\hat{g})^T A(g-\hat{g})}\, dg = e^{\Psi(\hat{g})}(\det 2\pi A^{-1})^{1/2}.$$

We can approximate the log marginal likelihood as

$$\log q(c,k) = \Psi(\hat{g}) - \frac{1}{2}\log\det A + \frac{k}{2}\log 2\pi. \tag{4.1}$$

4.2.1. FINDING THE LOCATION OF THE MAXIMUM

To find expressions to find the location of the maximum $\hat{g}$ and the negative Hessian matrix A, we can compute the gradient of Ψ as

$$\nabla\Psi(g) = \nabla\log p(Y \mid c,k) - e^c\Lambda^q g$$

and the Hessian matrix as

$$H\Psi(g) = H\log p(Y \mid c,k) - e^c\Lambda^q.$$

We write explicit expressions for the gradient and the Hessian for both the probit and the logit model in terms of the link function h with derivative h'.

Lemma 4.2.1. *In both the probit and the logit case, we can write the Hessian matrix $H\Psi(g)$ as $-U^TWU$ for a diagonal matrix W. It is negative definite, so a unique maximum of $\Psi(g)$ exists.*

Proof. In the probit model, we use the properties that $\partial(Ug)_i/\partial g_j = u_{i,j}$, $\phi(x) = \phi(-x)$ and $\phi'(x) = -x\phi(x)$ to see that

$$\frac{\partial\log\Phi(Y_i(Ug)_i)}{\partial g_j} = \frac{Y_i\phi((Ug)_i)u_{i,j}}{\Phi(Y_i(Ug)_i)}$$

and

$$\frac{\partial^2 \log \Phi(Y_i(Ug)_i)}{\partial g_j \partial g_l} = -\left(\frac{\phi((Ug)_i)^2}{\Phi(Y_i(Ug)_i)^2} + Y_i(Ug)_i \frac{\phi((Ug)_i)}{\Phi(Y_i(Ug)_i)}\right) u_{i,j} u_{i,l}.$$

In the logit model, we use that $\partial(Ug)_i/\partial g_j = u_{i,j}$, $h'(x) = (1-h(x))h(x)$ and $h'(x) = h'(-x)$ to see that

$$\frac{\partial \log h(Y_i(Ug)_i)}{\partial g_j} = Y_i(1 - h(Y_i(Ug)_i))u_{i,j}$$

and

$$\frac{\partial^2 \log h(Y_i(Ug)_i)}{\partial g_j \partial g_l} = -h((Ug)_i)(1 - h((Ug)_i))u_{i,j}u_{i,l}.$$

It is directly seen that the resulting diagonal matrix W has positive entries in both cases. □

We can find the maximum of $\Psi(g)$ by Newton's method. The update step in Newton's method is given by

$$\begin{aligned} g^{\text{new}} &= g - (H\Psi)^{-1}\nabla\Psi \\ &= g + (U^T WU + e^c \Lambda^q)^{-1}(\nabla \log p(Y \mid g, c, k) - e^c \Lambda^q g). \end{aligned}$$

Instead of computing the inverse of $H\Psi(g)$ and its determinant directly, we use the matrix

$$B = I + e^{-c}\Lambda^{-q/2}U^T WU\Lambda^{-q/2} \tag{4.2}$$

for numerical stability, similar to the approach in Rasmussen and Williams (2006). The eigenvalues of B are bounded from below by 1 and, in case of a logit link function, bounded from above by $1 + e^{-c}n\lambda_1^{-q}/4$. In case of a probit link function, we observe that $\phi(x)/\Phi(-x) \to \infty$ as $x \to \infty$. The matrix inversion in the Newton procedure can be written as

$$(U^T WU + e^c \Lambda^q)^{-1} = e^{-c}\Lambda^{-q/2}B^{-1}\Lambda^{-q/2}$$

and the determinant as

$$\det A = e^c \prod_{j=1}^{k} \lambda_j^q \det B.$$

We can write Equation (4.1) as

$$\log q(c,k) = \sum_{i \in \text{obs}} \log h(Y_i(U\hat{g})_i) - \frac{1}{2}\log\det B - \frac{e^c}{2}\sum_{j=1}^{k} g_j^2 \lambda_j^q. \tag{4.3}$$

The method described in this section is summarized in Algorithm 4.1 below:

Algorithm 4.1 Newton's method for finding $\hat{g}$.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, initial values for paramter $g$ and hyperparameters $c$ and $k$.
Output: Mode for Laplace approximation $\hat{g}$ and the approximated log marginal likelihood $q(c,k)$.
1: repeat
2:     Compute $W$ as in Lemma 4.2.1.
3:     Compute $B$ as in Equation (4.2).
4:     Compute the Cholesky decomposition $B = LL^T$.
5:     Update $g \leftarrow g + B^{-1}\nabla\Psi(g)$ using $L$.
6: until convergence.
7: Compute $\log\det B$ using $L$.
8: Compute $\log q(c,k)$ as in Equation (4.3).
```

The most expensive step is the computation of B being of order $O(k^2 n)$. The Cholesky decomposition is of order $O(k^3)$, which is less expensive if k is much smaller than n.

4.3. OPTIMIZATION FOR HYPERPARAMETERS

To maximize the approximate log marginal likelihood, we optimize Equation (4.3) over the hyperparameters c and k. As Equation (4.3) is continuous in c and discrete in k, it is hard to simultaneously optimize over both hyperparameters. For fixed k however, the optimization over c is relatively straightforward, therefore, we propose to start at a low, fixed value of k, find the optimal c and increase k until the approximated log marginal likelihood doesn't increase significantly anymore. The pitfalls of this method are explored in Section 4.5. The bottom-up search is summarized in Algorithm 4.2 below:

Algorithm 4.2 Bottom-up search for k^*.

```
Input: Initial values for $c$ and $k$.
Output: $c^*$ and $k^*$
1: repeat
2:     Compute $c^*$ using Algorithm 4.4 or 4.5.
3:     Update $k \leftarrow k + 1$.
4: until convergence.
```

Note that the current c^* can be used as initial value in the next iteration. Also $\hat{g}$ can be used in the next iteration if it is for example appended by 0.

4.3.1. OPTIMIZATION OVER THE REGULARIZATION PARAMETER

If we fix the truncation level k, we can write an explicit expression for the derivative of the approximate log marginal likelihood with respect to c. One has to note that $\hat{g}$ and A also depend on the hyperparameter c. The derivative with respect to c can be computed using the chain rule as

$$\frac{d\log q(c,k)}{dc} = \frac{\partial \log q(c,k)}{\partial c} + \sum_{j=1}^{k} \frac{\partial \log q(c,k)}{\partial \hat{g}_j} \frac{\partial \hat{g}_j}{\partial c}. \tag{4.4}$$

The first term is given by

$$\frac{\partial \log(c,k)}{\partial c} = \frac{k}{2} - \frac{1}{2}\mathrm{tr}B^{-1} - \frac{e^c}{2} \sum_{j=1}^{k} \hat{g}_j{}^2 \lambda_j^q. \tag{4.5}$$

The partial derivative of $\hat{g}$ is given by

$$\frac{\partial \hat{g}}{\partial c} = -\Lambda^{q/2} B^{-1} \Lambda^{q/2} g. \tag{4.6}$$

The remaining partial derivatives are given by

$$\begin{aligned}\frac{\partial \log q(c,k)}{\partial g_j} &= -\frac{1}{2} \frac{\partial \log \det B}{\partial g_j} \\ &= \mathrm{tr}\left(B^{-1} \frac{\partial B}{\partial g_j}\right),\end{aligned} \tag{4.7}$$

where

$$\frac{\partial B}{\partial g_j} = e^{-c} \Lambda^{-q/2} U^T \frac{\partial W}{\partial g_j} U \Lambda^{-q/2}. \tag{4.8}$$

We can see that for the probit model

$$\begin{aligned}\frac{\partial W_{i,i}}{\partial g_j} = \Bigg(& -2(Ug)_i \frac{\phi((Ug_i))^2}{\Phi(Y_i(Ug)_i)^2} - 2Y_i \frac{\phi((Ug_i))^3}{\Phi(Y_i(Ug_i))^3} + Y_i \frac{\phi((Ug_i))}{\Phi(Y_i(Ug_i))} \\ & -(Ug)_i \frac{\phi((Ug_i))^2}{\Phi(Y_i(Ug_i))^2} - Y_i (Ug)_i^2 \frac{\phi((Ug_i))}{\Phi(Y_i(Ug_i))^2} \Bigg) u_{i,j}\end{aligned} \tag{4.9}$$

and for the logit model

$$\frac{\partial W_{i,i}}{\partial g_j} = -h((Ug)_i)(1 - h((Ug)_i))(2h((Ug)_i) - 1) u_{i,j}. \tag{4.10}$$

The computation of the derivative with respect to c is summarized in Algorithm 4.3 below:

Algorithm 4.3 Computation of the derivative $\frac{d \log q(c,k)}{dc}$.

Input: The input and results of Algorithm 1.
Output: The derivative of the approximated log marginal likelihood.

```
1: for j=1, ..., k do
2:     Compute $\frac{\partial W}{\partial g_j}$ as in Equation (4.9) or Equation (4.10).
3:     Compute $\frac{\partial B}{\partial g_j}$ as in Equation (4.8).
4:     Compute $\frac{\partial \log q(c,k)}{\partial \hat{g}_j}$ as in Equation (4.7).
5: end for
6: Compute $\frac{\partial \hat{g}}{\partial c}$ as in Equation (4.6)
7: Compute $\frac{\partial \log q(c,k)}{\partial c}$ as in Equation (4.5).
8: Compute $\frac{d \log q(c,k)}{dc}$ as in Equation (4.4).
```

The most expensive step is the computation of $\partial B/\partial g_j$. It is of order $O(k^2 n)$ and repeated k times resulting in a cost of $O(k^3 n)$.

For large k it might be cheaper to approximate the derivative by

$$\frac{\log q(c+\epsilon, k - \log q(c,k)}{\epsilon} \tag{4.11}$$

for small ϵ. In the successive calls to Algorithm 4.1, one can use the $\hat{g}$ of the first call as initial value in the second call.

With the derivative as an output of Algorithm 4.3 or by approximation, we can use a simple secant method (see, for example Section 9.2 of Press et al. (2007)) to find the optimal c^* for fixed k as summarized in Algorithm 4.4.

Algorithm 4.4 Secant method to find c^* for fixed k.

Input: Two initial values for c, say c_1 and c_2, fixed k.
Output: c^*.

```
1: repeat
2:     Use Algorithm 4.3 to compute $c_{t+2} \leftarrow c_{t+1} - \frac{\left.\frac{d\log q(c,k)}{dc}\right|_{c_{t+1}} (c_{t+1}-c_t)}{\left.\frac{d\log q(c,k)}{dc}\right|_{c_{t+1}} - \left.\frac{d\log q(c,k)}{dc}\right|_{c_t}}$.
3: until convergence.
```

Yet another approach to find c^* is to do a golden section search (see, for example Section 10.2 of Press et al. (2007)). The use of the golden ratio in Algorithm 4.5 ensures that function evaluations can be reused.

Algorithm 4.5 Golden section search to find c^* for fixed k.

```
Input: Initial initial interval for $c$, say $[a,b]$, fixed $k$, golden ration $\phi = (1+\sqrt{5})/2$.
Output: $c^*$
 1: repeat
 2:     Compute $x = b - (b-a)/\phi$.
 3:     Compute $y = a + (b-a)/\phi$.
 4:     if $\log q(x,k) > \log q(y,k)$ then
 5:         Update $(b, \log q(b,k)) \leftarrow (y, \log q(y,k))$.
 6:         Update $(y, \log q(y,k)) \leftarrow (x, \log q(x,k))$.
 7:         Update $x \leftarrow b - (a-b)/\phi$ and compute $\log q(x,k)$.
 8:     else
 9:         Update $(a, \log q(a,k)) \leftarrow (x, \log q(x,k))$.
10:         Update $(x, \log q(x,k)) \leftarrow (y, \log q(y,k))$.
11:         Update $y \leftarrow a + (b-a)/\phi$ and compute $\log q(y,k)$.
12:     end if
13: until convergence.
```

A closer look at the above algorithms can provide us some guidance to which one to use in which situation. We proposed three variants: using the exact derivative of the log marginal likelihood with respect to c (Algorithm 4.3), numerically approximating the derivative with respect to c (Equation (4.11)) and a golden section search (Algorithm 4.5), where we don't use the derivative at all. While all three methods will eventually find c^*, so we will not observe a difference in our classification outcome, they might differ in computational costs. Explicitly computing the derivative of the log marginal likelihood with respect to c as in Algorithm 4.3 has a cost of $O(k^3 n)$. Numerically computing the derivative by using two successive calls to Algorithm 4.1 has two times an iteration step of cost of order $O(k^2 n)$. The number of iterations in the second call can be reduced by starting in the final value of $\hat{g}$ of the first call. However, in both these variants we sometimes find that, if we initiate c far away from c^*, that the resulting step size is too large. It might happen that the next evaluation of the derivative with respect to c is in such a large value for c that it results in numerical errors. We can overcome these problems by assigning a maximum step size or by scaling the step size with a factor $\gamma \in (0,1)$. A method for one-dimensional optimization that is both conservative in the number of function iterations and does not require the computation of the derivative is golden section search. While it does not use the information of the derivative, it always finds an optimal c within the initial interval $[a,b]$, which should be chosen wide enough such that c^* is in there, but also making sure that prohibitive values of c causing numerical errors are excluded.

4.4. SAMPLING SCHEME

Once c^* and k^* have been determined, we can sample from the posterior $p(g \mid c^*, k^*, D)$. There are two ways in we can sample from the posterior. We could use a Gibbs sampler using a latent variable approach (cf. Albert and Chib (1993)) as described in Algorithm 4.6. In this case, we have an approximate sample from the exact posterior distribution. Alternatively, we could sample from the Laplace approximation directly (see Rasmussen and Williams (2006)) as in Algorithm 4.7. In this case we have an exact sample from an approximation of the posterior. Sampling from the Laplace approximation is preferable from a computational perspective, since the posterior distribution is approximated with a Gaussian distribution with mean $\hat{g}$ and covariance matrix A^{-1}, which are both computed in the optimization step. However, it may be a poor approximation to the true shape of the posterior.

Algorithm 4.6 Gibbs sampler for posterior distribution.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, $c^*$, $k^*$ and an initial value for $g$,.
Output: Gibbs sample from posterior $p(g \mid c^*, k^*, D)$.
1: repeat
2:     Compute $f \leftarrow Ug$.
3:     for $i = 1, \dots, n$ do
4:         Draw independent
               $z_i \sim \begin{cases} N(f_i, 1), & \text{if } i \notin I^{\text{obs}}, \\ N_+(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = 1, \\ N_-(f_i, 1), & \text{if } i \in I^{\text{obs}} \text{ and } y_i = -1. \end{cases}$
5:     end for
6:     Draw
               $g \sim N((I + e^c \Lambda^q)^{-1} U^T z, (I + e^c \Lambda^q)^{-1})$.
7: until you have a large enough sample.
```

Here N_+ (N_-) denotes a normal distribution with given mean and variance, conditioned to be positive (negative). The output $\hat{g}$ from Algorithm 4.1 can be used to initialize the sampler.

Algorithm 4.7 Laplace sampler for posterior distribution.

```
Input: Hyperparameters $c^*$ and $k^*$.
Output: Sample from approximated posterior.
1: Run Algorithm 4.1 using $c^*$ and $k^*$.
2: Draw
               $g \sim N(\hat{g}, A^{-1})$.
```

4.4.1. PERFORMANCE RELATIVE TO HIERARCHICAL BAYES

The hierarchical Bayes sampler in Hartog and Van Zanten (2019) has a cost of $O(kn)$ per iteration and Algorithm 4.1 has a cost of $O(k^2 n)$ per iteration. Both costs are after an initial cost of $O(kn^2)$ of computing the first k eigenvalues, however, the two cannot be compared directly as an iteration in a Newton-type algorithm is very different from an iteration in an MCMC-type algorithm. The actual time it takes to get an estimate depends mainly on the implementation and the choices made about convergence criteria. Once the hyperparameters c^* and k^* are fixed, a draw from the Laplace approximation of the posterior is simply a draw from a k^*-dimensional multivariate normal distribution. One conceptual aspect in favor of empirical Bayes is that the selection of the hyperparameters and the estimation of the soft label function are in two separate steps. As a result, we can decide, with a limited computational budget, to be a bit imprecise in the determination of the hyperparameters to save time. In the MCMC algorithm for the full hierarchical Bayesian approach, the hyperparameters are sampled together with the coefficients of the soft label function, so limiting your costs in the determination of the hyperparameters only is not possible. However, the MCMC algorithm is more flexible in terms of the prior distribution used for the hyperparameters and, as it also produces a posterior distribution for c and k, it is able to quantify uncertainty in these parameters, whereas in the empirical Bayes method, they are simply fixed at c^* and k^*.

4.5. NUMERICAL EXPERIMENTS

4.5.1. FINDING THE OPTIMAL REGULARIZATION PARAMETER

We use a simulated example to illustrate our algorithms. We start with a realization of the Watts-Strogatz model (Watts and Strogatz, 1998). The graph is obtained by first considering a ring graph of 1000 nodes. Then we loop through the nodes and uniformly rewire each edge with probability 0.25. We keep the largest connected component and delete multiple edges and loops resulting in the graph in Figure 4.1 with 848 nodes.

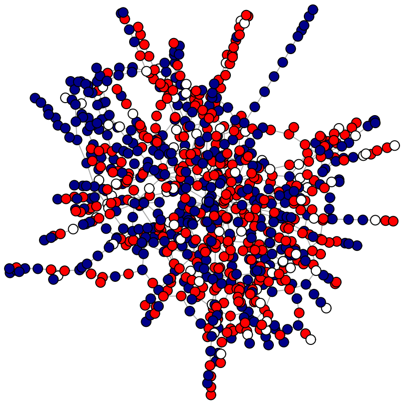


Figure 4.1: Small-world graph with two types of labels. The white vertices have unobserved labels.

We numerically determine the eigenvalues λ_j and eigenvectors $u^{(j)}$ of the Laplacian matrix and define a function f_0 on the graph by

$$f_0 = \sum_{j=1}^{n} a_j u^{(j)},$$

where we choose $a_j = \sqrt{n}(j-1)^{-2/r-1/2}\sin(j-1)$. This function has Sobolev-type smoothness $\beta = 2$ (cf. Kirichenko and Van Zanten (2017)). We assign labels to the vertices according to probabilities $P(Y_i = 1) = \Phi(f_0(i))$, where Φ is the distribution function of the standard normal distribution. We remove 10% of the labels.

For now, we fix k at 25 and inspect the corresponding approximated log marginal likelihood and its derivative in Figure 4.2. Note that $c \to \infty$ corresponds to an ever more concentrated prior on g around 0 and $c \to -\infty$ corresponds to ever more flatter prior. We observe that a Newton-type algorithm that starts with an initial value of c around 20 risks numerical errors by breaking too early or not finding the optimal $c^* = 5.8$ at all. A conservative choice might seem to take the initial value to be low, but an initial value around -10 will largely overshoot the optimal value and have a second iteration with a too large value for c. If no prior information about the order of magnitude of the optimal c^* is available, we suggest the safer choice of using golden section search with a wide starting interval.

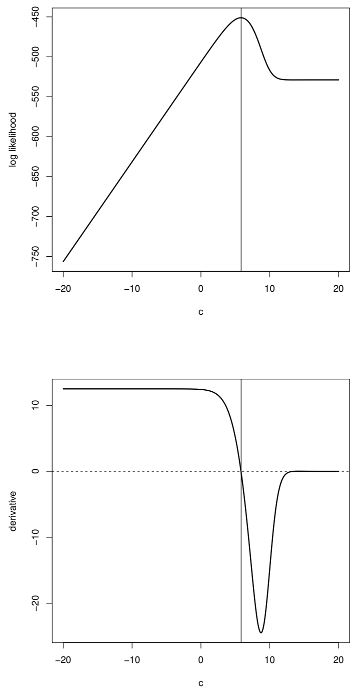


Figure 4.2: The approximated log likelihood in the small-world graph example for $k = 25$ as a function of c and its derivative.

4.5.2. FINDING THE OPTIMAL TRUNCATION LEVEL

The optimization over k is a discrete optimization problem, therefore we choose to split the joint optimization over c and k into its two separate parts. The optimization over c is a one-dimensional optimization over a continuous parameter. As the computational costs of this step increase with k (Algorithm 4.1 costs $O(k^2 n)$ per iteration), we propose a bottom-up search, where we start at a low value of k, compute the optimal c^* for this fixed k, and then increase k (Algorithm 4.2). We include one more eigenvector to the estimate of f in every iteration. At some point, adding an additional eigenvector as explanatory variable does not increase the log marginal likelihood significantly, and this is where we will stop the algorithm. We find that there are two drawbacks to this method. One drawback is that that it exhaustively searches all values of k from a low value of k up to k^*. An advantage of this method as compared to different search orders in k is that we can use the previous $\hat{g}$ and c as our initialization in our current k to reduce the number of iterations in Algorithm 4.1. Another problem can be that the log marginal likelihood has very small increments for low values of k followed by a large increment later on, but we might never get there. So we can trust our method only if we believe that the unknown function we try to estimate actually has a low dimensional representation in terms of eigenvectors of the Laplacian matrix. If this is not the case however, a good approximation by our proposed method might even be computationally prohibitive all together on account of the scaled computational costs with respect to k.

To illustrate our method, we use the small-world example from the previous subsection. In the left plot in Figure 4.3, we see the approximate log likelihood as a function of c for different values of k. In the middle plot we see the maximal approximate log marginal likelihood for different values of k. Note that we plotted only up to $k = 100$, whereas we could in theory have included all 848 possible values for k. However, we observe that the maximal approximate log marginal likelihood is achieved as early as $k = 3$ in this example, because our chosen function is very smooth. In the right plot we see, for different values of k, the corresponding optimal c. We observe that the optimal c increases with k. This makes sense, because having fewer terms in the prior (low values of k) makes it smoother, whereas low values for c make the prior rougher, so these two parameters balance each other out.

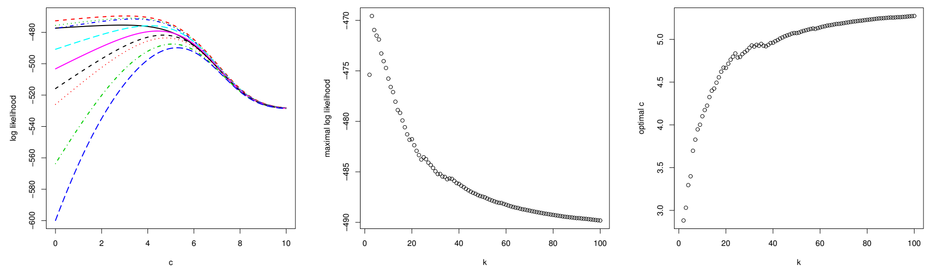


Figure 4.3: Left: The approximate log marginal likelihood for $k = 2,3,4,5,10,15,20,25,50,100$ as a function of c. Middle: The maximal approximate log marginal likelihood for different values of k. Right: The corresponding optimal value c^* for different values of k.

Bottom-up search might break down when the maximal approximate log marginal likelihood has a sudden large increase after a certain eigenfunction is included, because the search might have stopped before that point. A simple example is the function on the path graph shown in the left plot of Figure 4.4. In the right plot, we see that the maximal approximate marginal log likelihood makes a jump at $k = 25$. If we terminated our search before this point we would set $k^* = 3$. The two corresponding $\hat{g}$'s are in Figure 4.5, these correspond to a smooth and a rough explanation of the data. In this example the rough explanation would be the better one according to the approximate log marginal likelihood. However one can also think of a multimodal example, where both, or even multiple interpretations of the data are equal in terms of log marginal likelihood. In that case, it would be advisable to take the lowest k for computational reasons, but also for model simplicity.

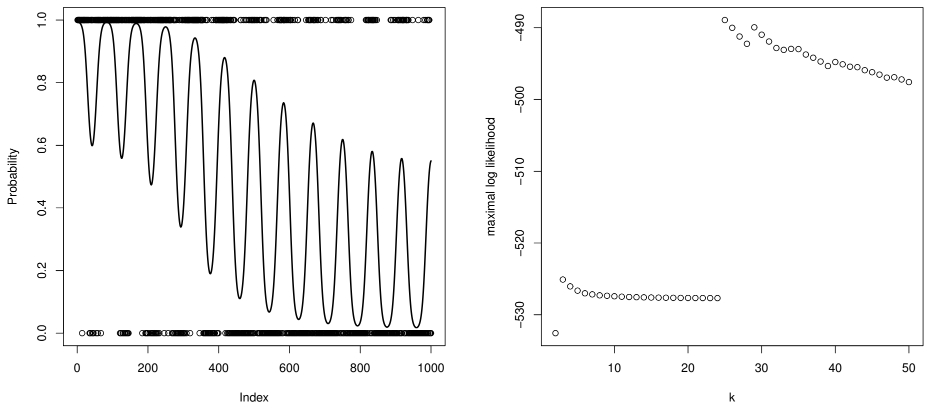


Figure 4.4: Left: A simulated example on the path graph with $n = 1000$. The circles are the observations and the black line is the true soft label function. Right: The maximal approximate log marginal likelihood for different values of k.

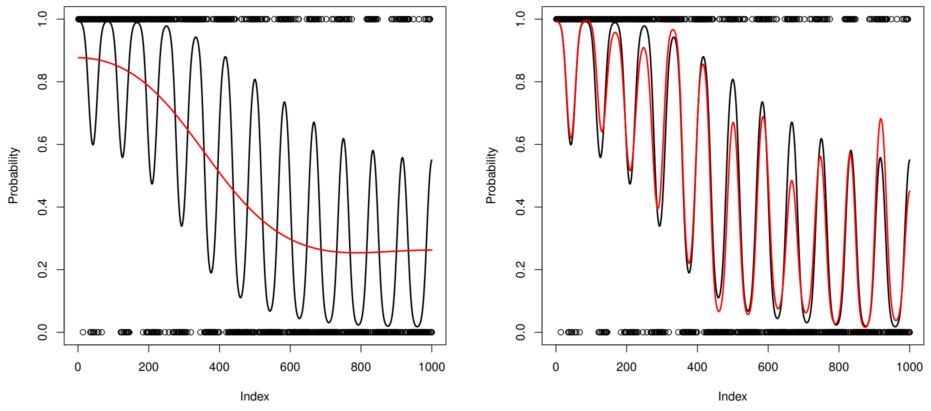


Figure 4.5: Left: The left plot in Figure 4.3, with the soft label function estimate $\hat{\ell} = \Phi(U\hat{g})$ for $k = 3$ (red). Right: The left plot in Figure 4.3 with soft the label function estimate $\hat{\ell} = \Phi(U\hat{g})$ for $k = 25$ (red).

4.5.3. MCMC VERSUS LAPLACE APPROXIMATION

In Section 4.4, we propose two methods to sample from the posterior distribution once we have found c^* and k^*. Algorithm 4.6 is a Gibbs sampler using a latent variable approach as in Albert and Chib (1993). This is relevant to the probit link function. It is generally more involved than Algorithm 4.7, where we sample from the Laplace approximation. The Laplace approximation of the posterior is a Gaussian distribution, so this allows for simple direct sampling. A drawback of the Laplace approximation is that a Gaussian distribution might not be a good approximation to the true posterior. This can be the case if, for example, the true posterior is multimodal or if the true posterior doesn't have elliptical joint marginal densities. In that case, we also expect the Gibbs sampler to suffer.

In our small-world graph example, the quality of the estimate and intervals of the Laplace approximation is the same as of an MCMC sample as shown in Figure 4.6. We also don't observe any qualitative differences between the correlations plots of the samples for the first five points, as shown in Figure 4.7. This suggests that a Gaussian distribution is a good enough approximation to the true posterior distribution.

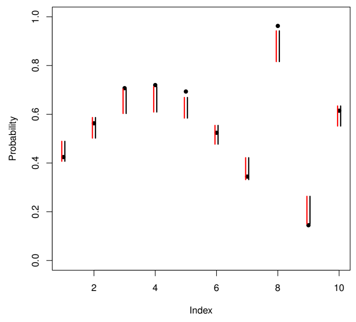


Figure 4.6: Estimated 95% credible intervals for the soft label function evaluated at ten of the missing points based on a sample of 3000 draws from the posterior using the Laplace approximation (red) and an MCMC sampler (black). The black dots are the true soft label function values.

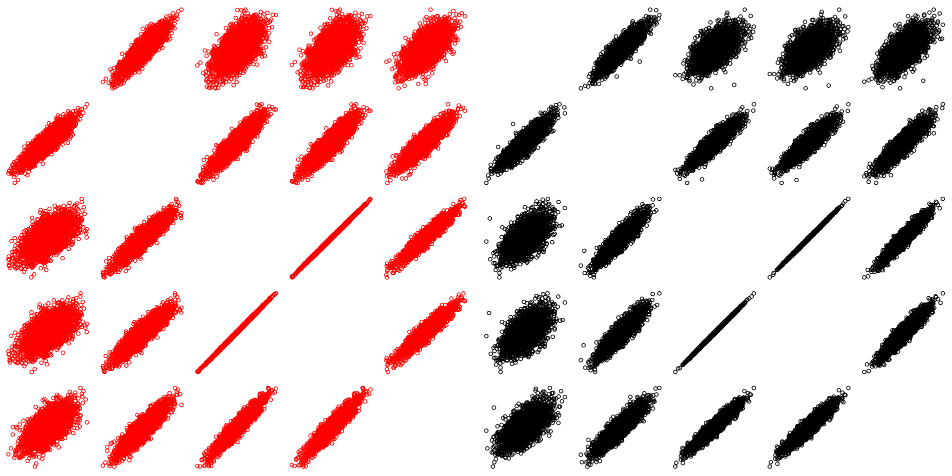


Figure 4.7: Correlation plots of the samples from the posterior for the first five points from Figure 4.6. Laplace approximation is in red and the MCMC sample in black.

4.5.4. CHANGING OBJECT IN A NOISY ENVIRONMENT

In the example in this section, we apply our method to a large graph constructed in the following way. As a ground truth we use the following animation of a morphing object. The animation consists of 50 frames of 50×50 pixels. The labels on each pixel indicate whether the pixel is of the object or not. We flip 10% of the labels at random. To convert this problem into a graph problem, we connect neighboring pixels in each frame and with the corresponding pixels in the previous and next frame, resulting in a $50 \times 50 \times 50$ grid graph on a total of $n = 125000$ nodes. We can explicitly compute the eigenvalues using the known eigenvalues of the path graph using Theorem 3.5 in Mohar (1991). The corresponding eigenvectors are given by the tensor products of the eigenvectors of the path graph and can also be computed explicitly. A selection of the noisy images are shown in Figure 4.8. In Figure 4.9, we see the estimated soft label function, which we can truncate at 0.5 to predict the shape of the object, see Figure 4.10.

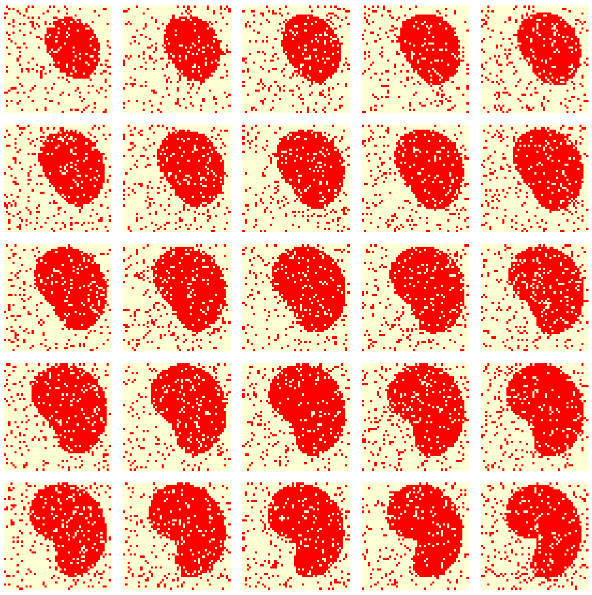


Figure 4.8: 25 frames of the true underlying morphing object with noise.

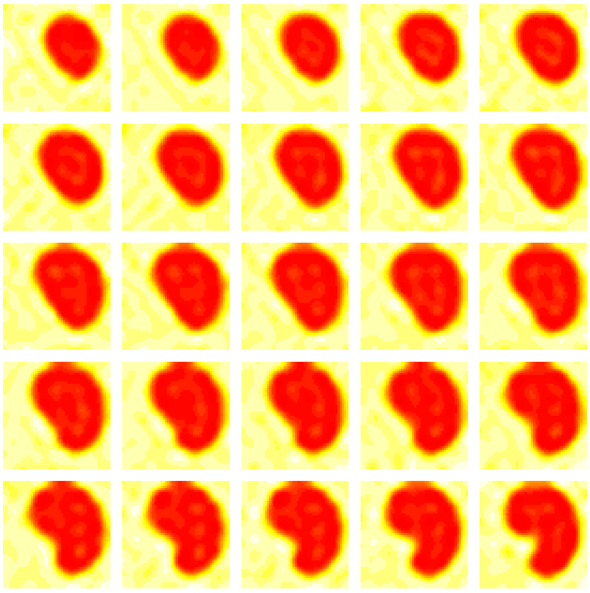


Figure 4.9: 25 frames of the estimated soft label function.

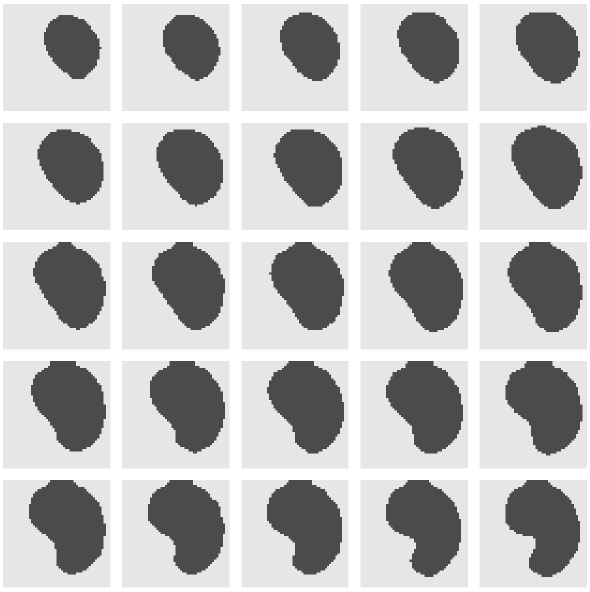


Figure 4.10: 25 frames of the estimated labels, using truncation at 0.5.

4.6. REGRESSION PROBLEM

In this section, we demonstrate the empirical Bayes approach to solve the regression problem on a graph. We repeat the regression setting for completeness. The graph is denoted by $G = (V, E)$, where $V = \{1, \dots, n\}$. We assume that the noisy labels y are Gaussian with

$$y \mid f \sim N(f, \sigma^2 I),$$

where f is the function of interest. For simplicity, we assume σ^2 is fixed and known and that all noisy label are observed. We are interested in inferring f from the data y. Similar to the classification problem in the previous sections, we consider a Gaussian prior on f, which depends on the Laplacian matrix L, on a multiplicative scale parameter $c > 0$ and a truncation level k. The resulting prior is given by truncated series

$$f = \sum_{j=1}^{k} g_j u^{(j)},$$

where

$$g \mid c, k \sim N(0, (e^c(\Lambda_k + n^{-2} I)^q)^{-1}),$$

for some fixed $q > 0$, where Λ_k is the restriction of the matrix of eigenvalues Λ, from the eigendecomposition $L = U\Lambda U^T$, to the first k rows and columns. The additional number n^{-2} is added to L to make it invertible as in Section 4.1. The empirical Bayes approach consists of finding c and k such that the marginal likelihood is maximized. The marginal likelihood is given by

$$p(y \mid c, k) \propto (\sigma^{-2})^{\frac{n}{2}} \left(\prod_{j=1}^{k} \frac{e^c(\lambda_j + n^{-2})^q}{\sigma^{-2} + e^c(\lambda_j + n^{-2})^q} \right)^{\frac{1}{2}} e^{-\frac{1}{2}\frac{y^T y}{\sigma^2} + \frac{1}{2}\sum_{j=1}^{k} \frac{(\sigma^{-2} y^T u^{(j)})^2}{\sigma^{-2} + e^c(\lambda_j + n^{-2})^q}},$$

so the log marginal likelihood is

$$\frac{1}{2}\sum_{j=1}^{k} \log(e^c(\lambda_j + n^{-2})^q) - \frac{1}{2}\sum_{j=1}^{k} \log(\sigma^{-2} + e^c(\lambda_j + n^{-2})^q) + \frac{1}{2}\sum_{j=1}^{k} \frac{(\sigma^{-2} y^T u^{(j)})^2}{\sigma^{-2} + e^c(\lambda_j + n^{-2})^q}$$

up to a constant. To maximize the log marginal likelihood, we propose a bottom-up search as in Algorithm 4.2, where in each iteration we find the optimal c^* using golden ratio search, Algorithm 4.5, or the secant method, Algorithm 4.4. The derivative of the log marginal likelihood with respect to the regularization parameter c is

$$\frac{k}{2} - \frac{1}{2}\sum_{j=1}^{k} \frac{e^c(\lambda_j + n^{-2})^q (\sigma^{-2} + e^c(\lambda_j + n^{-2})^q + (\sigma^{-2} y^T u^{(j)})^2)}{(\sigma^{-2} + e^c(\lambda_j + n^{-2})^q)^2}.$$

Given c^* and k^*, the posterior distribution of g is a normal distribution with mean

$$\sigma^{-2}(\sigma^{-2} I + e^c(\Lambda_k + n^{-2} I)^q)^{-1} U_k^T y$$

and variance

$$(\sigma^{-2} I + e^c(\Lambda_k + n^{-2} I)^q)^{-1}.$$

These steps can be summarized in the following algorithm:

Algorithm 4.8 Sampling scheme for the regression problem using empirical Bayes.

```
Input: Data $D = \{(i, y_i) : i = 1, \dots, n\}$, variance $\sigma^2$, initial values for $c$, $k$ and $g$.
Output: Hyperparameters $c^*$ and $k^*$, sample from the joint posterior $g \mid y$.
1: repeat
2:     Compute $c^*$ using Algorithm 4.4 or 4.5.
3:     Update $k \leftarrow k + 1$.
4: until convergence.
5: repeat
6:     Using $c = c^*$ and $k = k^*$ from the previous steps, sample
           $g \sim N(\sigma^{-2}(\sigma^{-2}I + e^c(\Lambda_k + n^{-2}I)^q)^{-1}U_k^T y, (\sigma^{-2}I + e^c(\Lambda_k + n^{-2}I)^q)^{-1})$.
7: until you have a large enough sample.
```

If there are any labels missing, we can adjust the log marginal likelihood to

$$\frac{1}{2}\sum_{j=1}^{k}\log(e^c(\lambda_j + n^{-2})^q) - \frac{1}{2}\log\det(\sigma^{-2}(U_k^{\text{obs}})^T U_k^{\text{obs}} + e^c(\Lambda_k + n^{-2}I)^q)$$
$$+\frac{1}{2}(y^{\text{obs}})^T U_k^{\text{obs}}(\sigma^{-2}(U_k^{\text{obs}})^T U_k^{\text{obs}} + e^c(\Lambda_k + n^{-2}I)^q)^{-1}(U_k^{\text{obs}})^T y^{\text{obs}},$$

where we can compute

$$((U_k^{\text{obs}})^T U_k^{\text{obs}})_{j,l} = \begin{cases} 1 - \sum_{i \in I^{\text{miss}}}(u_i^{(j)})^2, & j = l, \\ -\sum_{i \in I^{\text{miss}}} u_i^{(j)} u_i^{(l)}, & j \neq l, \end{cases}$$

if the number of missing labels is much smaller than the number of observed labels. The posterior distribution in line 6 of Algorithm 4.8 is now

$$g \sim N(\sigma^{-2}(\sigma^{-2}(U_k^{\text{obs}})^T U_k^{\text{obs}} + e^c(\Lambda_k + n^{-2}I)^q)^{-1}(U_k^{\text{obs}})^T y^{\text{obs}},$$
$$(\sigma^{-2}(U_k^{\text{obs}})^T U_k^{\text{obs}} + e^c(\Lambda_k + n^{-2}I)^q)^{-1}).$$

4.6.1. TRAFFIC FLOW ESTIMATION

To illustrate the regression problem, we consider a simulated example of traffic flow estimation inspired by Aldous and Shun (2010) and Bickel et al. (2007). Traffic data is collected by authorities to measure the performance of the road network in terms of flow, occupancy and speed at locations in the network. A common type of sensor used for traffic flow estimation is a *loop detector* that counts the number of passing vehicles over a certain time interval. However, the resulting loop data is often missing or invalid due to communication errors or malfunction. Missing and bad observations are imputed from the neighboring sensor location using a simple average or a regression approach.

We simulate a road network between cities based on a toy model. We sample 500 points uniformly on the unit square and compute the *relative neighborhood graph* by connecting nodes u and v if there does not exist a node w such that

$$\max\{\|u-w\|, \|v-w\|\} < \|u-v\|.$$

In Figure 4.11 this definition is illustrated, nodes u and v are connected if there is no node in the gray area. The resulting graph is simple and connected. Relative neighborhood graphs and related proximity graphs can be used as a toy models for road networks (see Aldous and Shun (2010)).

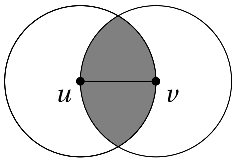


Figure 4.11: Construction of a relative neighborhood graph. Nodes u and v are connected if there is no point closer to both u and v as they are to each other, corresponding to the gray area not containing a node.

The resulting graph is the left graph in Figure 4.12. As traffic flow is a property of the roads in the city graph, we construct its *line graph*, by associating a vertex with each road in the city graph and connecting two vertices if the corresponding roads in the city graph have a city in common. This is the right graph in Figure 4.12 and is the road graph G we work with. It has $n = 617$ vertices.

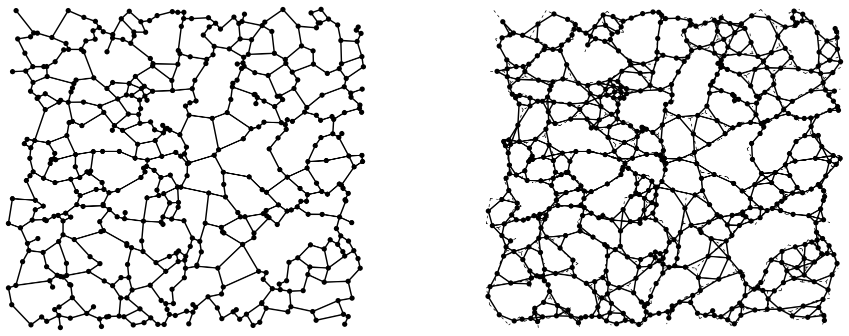


Figure 4.12: Left: A road network between cities as a realization of the relative neighborhood graph on 500 random uniform points on the unit square. Right: The corresponding line graph, where the nodes correspond to the roads of the left graph. The dashed line is the original graph.

We construct a true traffic flow function f_0 on the graph by setting

$$f_0 = \sum_{j=1}^{n} a_j u^{(j)},$$

where $u^{(j)}$ are the eigenvectors of the Laplacian matrix $L = U\Lambda U^T$ of G. To construct the coefficients a_j we consider the path graph with vertices $x_i = i/n$ for $i = 1,\dots,n$ and the function $5 + f(x_i)$, where f is given by Equation (1.8). We compute a_j as the coefficients of $5 + f$ on the path graph and use them in our graph G, generalizing the function on the path graph to our graph G. The resulting underlying traffic flow is depicted in Figure 4.13 on both the city graph and the road graph G.

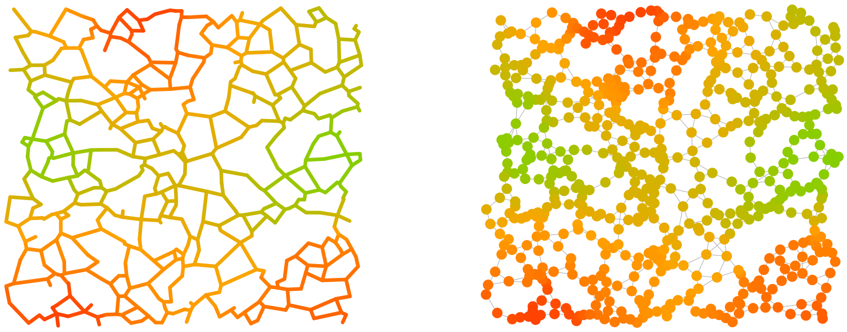


Figure 4.13: Left: The corresponding traffic flow on the city graph, where the flow is observed on the edges. Right: The traffic flow on the road graph G. Traffic intensity ranges from green (light traffic) to red (heavy traffic) and is observed on the nodes.

To construct a noisy observation of the traffic flow, we add independent noise $\epsilon_i \sim N(0,1)$ to each node $i = 1,\dots,n$ and remove 123 of our observations as depicted in Figure 4.14.

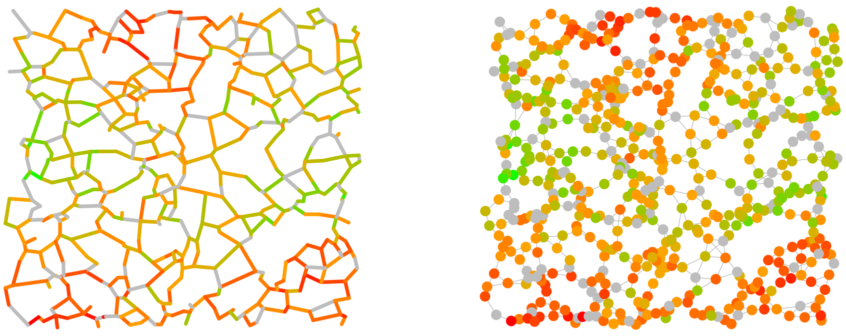


Figure 4.14: The noisy observation of the underlying traffic flow in Figure 2.24. The missing observations are in gray.

We estimate the traffic flow using Algorithm 4.8 with $q = 2$. We numerically determine $r = 1.67$ from the spectrum of the Laplacian matrix as shown in Figure 4.15. The resulting posterior mean for $f = Ug$ is shown in Figure 4.16. To illustrate the uncertainty in our estimate, we also plot the estimated traffic flow along a path from the north-west corner of the map to the south-east corner in Figure 4.17 together with point-wise 95% credible intervals. The optimal values for the hyperparameters are $c^* = 2.8634$ and $k^* = 10$.

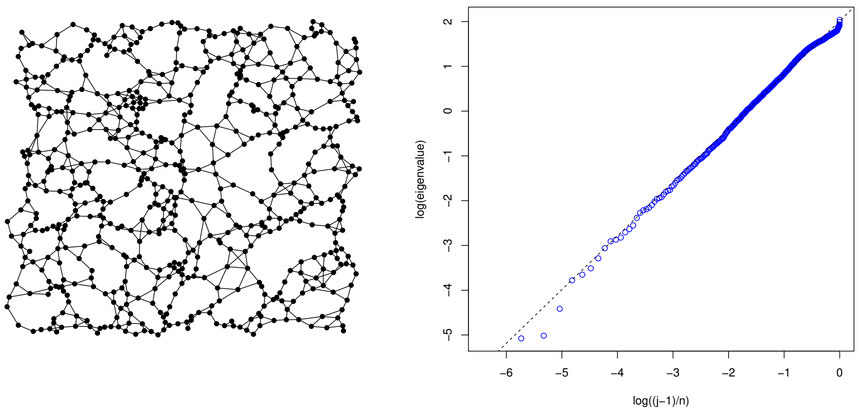


Figure 4.15: Left: The road graph. Right: The eigenvalues of the Laplacian matrix of the road graph. The dashed line corresponds to geometry number $r = 1.67$ based on a linear fit with slope 1.24. We have used the first 35% of the eigenvalues with exception of the first three.

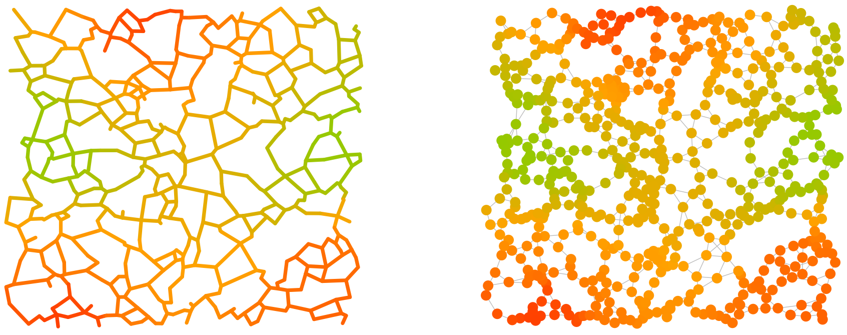


Figure 4.16: The posterior mean for the traffic flow data from Figure 4.14.

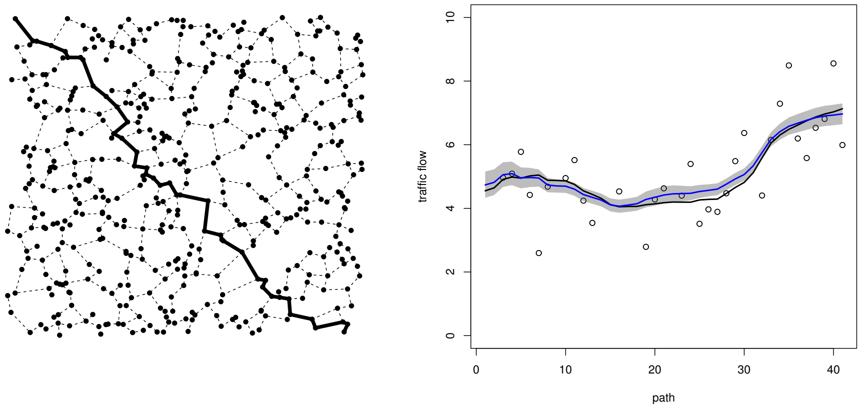


Figure 4.17: Left: A path in the city graph. Right: Posterior mean (blue) and point-wise 95% credible interval (gray area) for the traffic flow along the path. The points are the noisy observations and the black line is the true underlying traffic flow function.

4.7. CONCLUDING REMARKS

We have described a nonparametric Bayesian procedure to solve regression and binary classification problems on graphs. We have considered an empirical Bayes approach using a prior that can be represented as a scaled, truncated series of Gaussians. The proposed method consists of two steps: setting the hyperparameters and estimating the soft label function. To learn the regularization parameter and truncation level from the data, we use Laplace approximation in the classification problem and maximize the approximated marginal likelihood. After we have found and fixed hyperparameters we can easily sample from the posterior distribution, or the Laplace approximation to the posterior distribution. We have given details for an implementation of an efficient algorithm to perform these tasks. Our numerical experiments suggest that good results are obtained when using Algorithms 4.1 (finding the point in which to approximate), 4.5 (golden section optimization over c), 4.2 (bottom-up search over k) and 4.7 (sampling from the Laplace approximation). As Laplace approximation can be generalized to the case of multiple classes (see for example Rasmussen and Williams (2006)), we expect our proposed method to generalize to multiple classes as well.

5
Variational inference

In this chapter, we describe an implementation of variantional inference to solve binary classification problems on graphs. We consider a Gaussian prior that is constructed by truncating a series expansion of the soft label function using the eigenfunctions of the Laplacian matrix as basis functions. We approximate the posterior distribution with independent Gaussian distributions. We demonstrate coordinate ascent variational inference to find optimal values for the hyperparameters. As a variant to coordinate ascent variational inference, we consider stochastic variational inference to make the variational approach scalable to large data sets.

In this chapter, we consider an alternative approximation to the empirical Bayes approach from the previous chapter in the context of binary classification problems on graphs. The setup of the classification problem is the same as in Chapter 2, 3 and 4 and outlined in Section 1.4, but is included here for completeness. Classification problems on graphs arise in a variety of applications, for instance in machine learning (e.g. Belkin et al. (2004); Sindhwani et al. (2007)), in the prediction of the biological function of a protein in a protein-protein interaction graph (e.g. Kolaczyk (2009); Nariai et al. (2007); Sharan et al. (2007)), in image analysis (e.g. Liu et al. (2014)) and in the prediction of brand preference in social networks (Blair et al. (2003); Smith (2011)). For the problems we have in mind, the graph is given or constructed by the application context. In binary classification, the vertices of the graph can be of two different types. This in encoded in the vertex label which is either 1 or -1. We have noisy observations of the labels of part of the vertices of a large given graph. The goal is to classify all vertices correctly, including those for which there is no observation available. The idea is that typically, the location of a given vertex in the graph, in combination with (noisy) information about the labels of vertices close to it, should have predictive power for the label of the vertex of interest. Hence, successful prediction of labels should be possible to some degree.

The Bayesian approach in the previous chapters, Hartog and Van Zanten (2018) and Kirichenko and Van Zanten (2017) consists of endowing the label function f with a prior distribution and computing the posterior distribution. The posterior distribution for f

that results from a Bayesian analysis can be used for prediction. The prior we consider is a Gaussian prior, with covariance structure based on the Laplacian matrix of the graph. The Laplacian matrix is used to take the geometry of the graph into account (cf.Ando and Zhang (2007); Belkin et al. (2004); Hartog and Van Zanten (2018); Kolaczyk (2009); Zhu and Hastie (2005). Using the eigendecomposition of the Laplacian matrix, we can represent the prior as a random truncation of a series of Gaussian random variables

$$f = \sum_{j=1}^{k} g_j u^{(j)}, \tag{5.1}$$

where k is the truncation level, $u^{(j)}$ are the eigenvectors of the Laplacian matrix and g_j are independent Gaussian random variables for $j = 1, \ldots, k$ with a variance that depends on a hyperparameter c that allows to adjust the multiplicative scale of the prior. We have previously considered a full Bayesian treatment (Chapter 3) and an emprical Bayes method (Chapter 4) to determine the hyperparameters c and k and infer the posterior distribution of f.

In the present chapter, we use variational inference to approximate the posterior distribution with a family of Gaussian distributions. We use an iterative procedure to compute the parameters of approximating family of distributions. An advantage of the variational inference approach is that it uses deterministic updates similar to Gibbs sampling, but is usually faster (Blei et al., 2017). Additionally, we will use stochastic variational inference to take only a part of the observed data into account in each iteration (cf. Hoffman et al. (2013); Kushner and Yin (1997)). This will result in much faster iterations and therefore applicability of our method to very large graphs.

The rest of this chapter is organized as follows. In the next section, we describe the classification problem setting and the priors we consider. In Section 5.2, we describe the variational approach, the evidence lower bound that we maximize in order to approximate the posterior distribution. In Section 5.3 and Section 5.2, we give the coordinate ascent variational inference and stochastic variational inference algorithms to find a (local) maximum of the evidence lower bound. The stochastic variational inference algorithm uses stochastic optimization to trade a less accurate gradient for faster computations. Subsequently, in Section 5.5, we propose how to sample from the posterior distribution using the outcome of the algorithms. In Section 5.6, we give a simple example application of finding an object in a noisy animation, for a graph of over 200000 vertices, demonstrating the improved scalability of the stochastic variational inference approach.

5.1. OBSERVATIONAL MODEL AND PRIOR

5.1.1. OBSERVATIONAL MODEL

The context of our problem setup is the same as in Chapter 2 (Hartog and Van Zanten, 2018). We have a given connected, simple graph $G = (v, E)$ with $\#V = n$ vertices, denoted for simplicity by $V = \{1, \ldots, n\}$. Associated to every vertex i is a random hard label $y_i \in \{-1, 1\}$. We assume that the variables y_i are independent for $i = 1, \ldots, n$, so that their joint distribution is determined by the unobserved soft label function $\ell : V \to (0, 1)$ given by

$$\ell(i) = P(y_i = 1) = 1 - P(y_i = -1).$$

The observed data is $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, where $I^{\text{obs}} \subset V$ is drawn from an arbitrary distribution μ on the collection 2^V if subsets if vertices. The exact sampling mechanism μ is not important for the algorithm we propose, only that the subset is independent of the labels. Throughout, we use the well-known latent variable perspective on this model (cf. Albert and Chib (1993)). This is simply the observation that we can sample Bernoulli variables $y_1, \dots, y_n$ with success probabilities $\ell(1), \dots, \ell(n)$ using an intermediate layer of latent Gaussian variables. Indeed, let Φ be the probit link function, i.e. the cdf of the standard normal distribution. Then, if $f : V \to \mathbb{R}$ is given, sampling independent Bernoulli variables y_i with success probabilities $\ell(i) = \Phi(f(i))$ can be achieved by subsequently sampling independent Gaussian variables z_i with mean $f(i)$ and variance 1 and then setting $y_i = 1_{z_i > 0}$ for $i = 1, \dots, n$.

5.1.2. PRIOR ON THE SOFT LABEL FUNCTION

To achieve a form of Bayesian Laplacian regularization in this problem, we put a Gaussian prior on the function f that determines the distribution of the hard labels, with a precision (inverse covariance) matrix given by a power of the Laplacian matrix L associated to the graph G. Recall that the Laplacian matrix is given by $L = D - A$, where D is the diagonal matrix of vertex degrees and A is the adjacency matrix of the graph. It is a symmetric, non-negative definite matrix. Since zero is always an eigenvalue of the Laplacian matrix (see Chapter 2), it is not invertible. To make the matrix invertible and thus suitable as a precision matrix, we add a small number $1/n^2$ to the diagonal. This number is motivated by the result that the smallest nonzero eigenvalue of the Laplacian matrix is at least $4/n^2$ (Theorem 4.2 of Mohar (1991)). We denote the eigenvalues fo the Laplacian matrix by $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_n$. As in Chapter 3 and Chapter 4, we use the eigendecomposition $L = U\Lambda U^T$, where Λ is the diagonal matrix of eigenvalues, and U the orthonormal matrix of corresponding eigenvectors, to write the function $f = Ug$ as a series expansion over the eigenvectors of the Laplacian matrix. We truncate this series at a random point k. We scale the series by a random scale parameter c and define

$$f = \sum_{j=1}^{k} g_j u^{(j)},$$

where $u^{(j)}$ is the jth eigenvector of the Laplacian matrix L, and

$$g \mid c, k \sim N(0, (e^c(\Lambda_k + n^{-2}I)^q)^{-1}),$$

where Λ_k is the restriction of Λ to the first k rows and columns.

5.1.3. PRIOR ON THE REGULARIZATION PARAMETER

We use the natural choice of prior for c, which is a gamma prior with density

$$p(c) \propto c^{a-1} e^{-bc}, \quad c > 0$$

for certain $a, b > 0$. This choice is motivated by the normal-inverse gamma partial conjugacy (see e.g. Choudhuri et al. (2007); Liang et al. (2007) in the context of our setting) and the positive results in the numerical experiments in our previous chapters. We can even choose the improper prior corresponding to $a = b = 0$, in which case $p(c) \propto 1/c$ as in Choudhuri et al. (2007); Hartog and Van Zanten (2018)) or $a = 1$ and $b = 0$, such that $p(c) \propto 1$.

5.1.4. FULL HIERARCHICAL MODEL

The resulting model is

$$\begin{aligned} D &= \{(i, y_i) : i \in I^{\mathrm{obs}}\}, \\ I^{\mathrm{obs}} &\sim \mu, \\ y_i &= 1_{z_i > 0} \quad i = 1, \ldots, n, \\ z \,|\, f &\sim N(f, I), \\ f &= \sum_{j=1}^{k} g_j u^{(j)}, \\ g \,|\, c, k &\sim N(0, (c(\Lambda_k + n^{-2} I)^q)^{-1}), \\ c &\sim \Gamma(a, b). \end{aligned} \tag{5.2}$$

We consider the power of the Laplacian matrix $q \geq 0$ given. Note that we also use the notation q for the variational density below. It should be clear from the context when this is the case. Our goal is to compute $f \,|\, D$ and use it to predict the unobserved labels. Note that we have fixed the truncation level k for now. We will set this parameter using an empirical Bayesian method, similar to Chapter 4, so for the variational inference part, we can consider k fixed for the next two sections.

5.2. VARIATIONAL APPROXIMATION

Variational inference approximates the posterior distribution $p(c, g, z \,|\, D)$ of our unobserved parameters c, g, z given the observations D, with a family of *variational distributions* $q(c, g, z)$ (Blei et al., 2017). A common assumption for the variational distribution is that its marginal distributions over the different parameters are independent (see, e.g. Damianou et al. (2011); Titsias and Lawrence (2010) in the context of Gaussian processes). This implies that the variation density $q(c, g, z)$ is the product of variational densities for each parameter. In machine learning literature, this is called the *mean-field variational family* (Blei et al., 2017). In our case, we assume that the variational distribution can be factorized as

$$q(c, g, z) = q(c) \prod_{j=1}^{k} q_j(g_j) \prod_{i=1}^{n} q_i(z_i). \tag{5.3}$$

This is an approximation, because in the actual posterior distribution, the parameters are not independent, and the variational distributions do not have to be from the same family as the posterior distribution. In our case, we choose

- $q(c)$ to be a gamma density with scale κ and rate θ,
- $q_j(g_j)$ a normal density with mean μ_j and variance σ_j^2 and

- $q_i(z_i)$ a normal density with mean ν_i and variance τ_i^2, conditioned to have the same sign as y_i. If y_i is unobserved, z_i can have any sign.

These choices are motivated by the partial conjugacy between these distributions and are natural choices in the context of Gaussian distributions (see, e.g. Blei et al. (2017); Damianou et al. (2011); Titsias and Lawrence (2010)). Within the variational families, our goal is to find the best approximation to the actual posterior distribution in terms of *Kullback-Leibler divergence*

$$\mathrm{KL}(q(c,g,z),p(c,g,z\,|\,D)) = \iiint \log \frac{q(c,g,z)}{p(c,g,z\,|\,D)} q(c,g,z)dcdgdz.$$

If we write expectation with respect to $q(c,g,z)$ as E_q, we can express the Kullback-Leibler divergence as

$$\mathrm{KL}(q(c,g,z),p(c,g,z\,|\,D)) = E_q \log q(c,g,z) - E_q \log p(c,g,z\,|\,D).$$

The Kullback-Leibler divergence is not symmetric in its two arguments. It is non-negative and is equal to zero if and only if its arguments are equal. It is not a feasible object function, because we wish to approximate the posterior $p(c,g,z\,|\,D)$ in the first place, so we cannot use that density to compute the Kullback-Leibler divergence. Instead, the alternative *evidence lower bound (ELBO)*

$$\begin{aligned}\mathrm{ELBO}(q(c,g,z)) &= \iiint \log \frac{p(c,g,z,D)}{q(c,g,z)} q(c,g,z)dcdgdz \\ &= E_q \log p(c,g,z,D) - E_q \log q(c,g,z) \\ &= E_q \log p(c,g,z\,|\,D) + \log p(D) - E_q \log q(c,g,z) \\ &= \log p(D) - \mathrm{KL}(q(c,g,z),p(c,g,z\,|\,D))\end{aligned}$$

is maximized (cf. Blei et al. (2017)). The last equation shows that ELBO is equal to the negative Kullback-Leibler divergence up to a constant $\log p(D)$ that doesn't depend on $q(c,g,z)$. Therefore, maximizing ELBO and minimizing the Kullback-Leibler divergence is equivalent. The name evidence lower bound comes from the result that it is a lower bound for the log evidence $\log p(y)$ (Jordan et al., 1999).

5.2.1. EVIDENCE LOWER BOUND

We maximize the ELBO over the hyperparameters ν, μ, σ^2 and θ. In this subsection, we will write the ELBO as a function of these hyperparameters. We derive, for our setting, that

$$\begin{aligned}\mathrm{ELBO}(\nu,\mu,\sigma^2,\theta) = E_q \log p(y\,|\,z) + \sum_{i=1}^{n} E_q \log p(z_i\,|\,g) + \sum_{j=1}^{k} E_q \log p(g_j\,|\,c) + E_q \log p(c) \\ - \sum_{i=1}^{n} E_q \log q_i(z_i) - \sum_{j=1}^{k} E_q \log q_j(g_j) - E_q \log q(c).\end{aligned} \tag{5.4}$$

We find an expression for each of the terms below.

The first terms $E_q \log p(y \mid z)$ ensures that y_i and z_i have the same sign if y_i is observed. Otherwise, z can have any value. The second term is given by

$$E_q \log p(z_i \mid g) = -\frac{1}{2}\log 2\pi - \frac{1}{2}E_q z_i^2 + E_q z_i \sum_{j=1}^{k} u_{i,j} E_q g_j - \frac{1}{2}\sum_{j=1}^{k} u_{i,j}^2 E_q g_j^2$$

and the third term is a sum of

$$E_q \log p(g_j \mid c) = -\frac{1}{2}\log 2\pi + \frac{1}{2}\log(\lambda_j + n^{-2})^q + \frac{1}{2}E_q \log c - \frac{1}{2}(\lambda_j + n^{-2})^q E_q c E_q g_j^2.$$

The fourth term is

$$E_q \log p(c) = a \log b - \log\Gamma(a) + (a-1)E_q \log c - bE_q c.$$

Furthermore, we have

$$E_q \log q_i(z_i) = \begin{cases} -\frac{1}{2}\log 2\pi - \frac{1}{2}\log\tau_i^2 - \frac{1}{2}\frac{E_q(z_i - \nu_i)^2}{\tau_i^2}, & i \notin I^{\text{obs}}, \\ -\frac{1}{2}\log 2\pi - \frac{1}{2}\log\tau_i^2 - \frac{1}{2}\frac{E_q(z_i - \nu_i)^2}{\tau_i^2} - \log\Phi\left(y_i \frac{\nu_i}{\tau_i}\right), & i \in I^{\text{obs}}, \end{cases}$$

$$E_q \log q_j(g_j) = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log\sigma_j^2 - \frac{1}{2}\frac{E_q(g_j - \mu_j)^2}{\sigma_j^2}$$

and

$$E_q \log q(c) = \kappa \log\theta - \log\Gamma(\kappa) + (\kappa - 1)E_q \log c - \theta E_q c.$$

The necessary moments are standard moments for the normal and gamma distributions and can be calculated as

$$E_q z_i = \begin{cases} \nu_i, & i \notin I^{\text{obs}}, \\ \nu_i + y_i \frac{\phi(\nu_i)}{\Phi(y_i \nu_i)}, & i \in I^{\text{obs}}, \end{cases} \tag{5.5}$$

$$E_q z_i^2 = \begin{cases} 1 + \nu_i^2, & i \notin I^{\text{obs}}, \\ 1 + \nu_i^2 + y_i \nu_i \frac{\phi(\nu_i)}{\Phi(y_i \nu_i)}, & i \in I^{\text{obs}}, \end{cases} \tag{5.6}$$

$$E_q g_j = \mu_j, \tag{5.7}$$

$$E_q g_j^2 = \mu_j^2 + \sigma_j^2, \tag{5.8}$$

$$E_q c = \frac{a + k/2}{\theta}, \tag{5.9}$$

$$E_q \log c = \psi(a + k/2) - \log\theta, \tag{5.10}$$

where $\psi(\cdot)$ is the digamma function, defined as the derivative of $\log\Gamma(\cdot)$. We have also used that $\tau_i = 1$ and $\kappa = a + k/2$ as we will deduce in the next section. Note that $E_q(g_j - \mu_j)^2 = \sigma_j^2$, $E_q(z_i - \nu_i)^2 = 1$ for $i \notin I^{\text{obs}}$ and $1 - y_i\nu_i\phi(\nu_i)/\Phi(y_i\nu_i)$ for $i \in I^{\text{obs}}$. Now that we have written the ELBO for our specific model (Equation (5.4)) as a function of the hyperparameters ν, μ, σ^2 and θ, our next goal is to find the optimal hyperparameter, such that the ELBO is maximized.

5.3. COORDINATE ASCENT VARIATIONAL INFERENCE

To maximize the ELBO over the hyperparameters of q in a mean-field family, we can use *coordinate ascent variational inference (CAVI)* (Bishop, 2006). This method updates the hyperparameters for each factor in Equation (5.3) iteratively, fixing all the other hyperparameters in each iteration. This approach is based on the lemma below. For a mean-field density $q(\xi) = \prod_i q_i(\xi_i)$, we introduce the notation $E_{-\xi_i}$ for the expectation with respect to the density $\prod_{j\neq i} q_j(\xi_j)$. Note that in our setting $\xi = (c, g, z)$.

Lemma 5.3.1. *Let $q(\xi) = \prod_i q_i(\xi_i)$ be a mean-field approximation to the actual posterior $p(\xi \mid D)$. If we fix $q_j(\xi_j)$ for all $j \neq i$, the optimal choice, with respect to maximizing ELBO, for $q_i(\xi_i)$ is proportional to*

$$\exp\left(E_{-\xi_i} \log p(\xi, D)\right). \tag{5.11}$$

Proof. This proof replicates the derivation in Blei et al. (2017), a different argument can be found in Bishop (2006). Using iterated expectations and the mean-field factorization, we can write the ELBO as

$$\text{ELBO} = E_{\xi_i} E_{-\xi_i} \log p(\xi, D) - E_{\xi_i} \log q_i(\xi_i) - \sum_{j\neq i} E_{\xi_j} \log q_j(\xi_j).$$

The last term is constant with respect to ξ_i. We see that the ELBO is equal to the negative Kullback-Leibler divergence between $q_i(\xi_i)$ and expression (5.11), up to a constant. Therefore the ELBO is maximized if $q_i(\xi_i)$ is proportional to (5.11). Note that in $p(\xi, D)$, the only variable is ξ_i, as all the other ξ_j for $j \neq i$ are fixed. □

Iteratively updating each factor $q_i(\xi_i)$ results in a monotone increasing sequence of values of the ELBO and will therefore converge to a local maximum. As the ELBO is not convex, there is no guarantee to find a global maximum. A practical solution is to consider multiple initial values. Note that this method results in similar updates as in the Gibbs sampler (Algorithm 1.1). Below, we find the optimal variational updates in our setting using Equation (5.11).

5.3.1. HYPERPARAMETER UPDATES

In our setting, we have chosen conjugate variational densities, so that each update (cf. Equation 5.11) only entails an update of the hyperparameters. Similar to the computation of conjugate conditionals in the Gibbs sampler (Chapter 2 and Chapter 3), the functional form of Equation (5.11) shows us the optimal variational distribution.

To illustrate how to use expression (5.11) to see that the optimal update for the variational distribution is just an update of the hyperparameters, we will detail this for the update of $q_i(z_i)$. Expression (5.11) implies that the optimal log variational density is equal to

$$E_{-z_i} \log p(c, g, z, D)$$

up to a constant. The expectation is with respect to $q(c)\prod_{j=1}^k q_j(g_j)\prod_{l\neq i} q_l(z_l)$. Now, for the log joint density

$$\log p(c, g, z, D) = \log p(y_i \mid z_i) + \sum_{l\neq i} \log p(y_l \mid z_l) + \log p(z_i \mid g) + \sum_{j=1}^{k} \log p(g_j \mid c) + \log p(c)$$

we see that the only part that depends on z_i is $\log p(y_i \mid z_i) + \log p(z_i \mid g)$. The first part ensures that y_i and z_i have the same sign, if y_i is observed. The second part is

$$\log p(z_i \mid g) = -\frac{1}{2}\log 2\pi - \frac{1}{2}\left(z_i - \sum_{j=1}^{k} u_{i,j} g_j\right)^2,$$

where we can expand the square to see that

$$E_{-z_i} \log p(z \mid g) = -\frac{1}{2}\log 2\pi - \frac{1}{2} z_i^2 + z_i \sum_{j=1}^{k} u_{i,j} E_{-z_i} g_j + E_{-z_i}\left(\sum_{j=1}^{k} u_{i,j} g_j\right)^2.$$

The first and last term do not depend on z_i, so the optimal log variational density for z_i in our context is equal to

$$-\frac{1}{2} z_i^2 + z_i \sum_{j=1}^{k} u_{i,j} E_{-z_i} g_j,$$

up to a constant, conditional on z_i having the same sign as y_i, if y_i is observed. We recognize that this has the similar functional form in z_i as the log density of a normal distribution with mean $\sum_{j=1}^{k} u_{i,j} E_{-z_i} g_j$ and variance 1. We can conclude that the optimal variational density for z_i is a normal distribution with the mean $\sum_{j=1}^{k} u_{i,j} E_{-z_i} g_j$ and variance 1, conditioned to have the same sign as y_i, if y_i is observed. This is exactly the variational distribution $q_i(z_i)$ we had chosen with hyperparameters ν_i and τ_i. Moreover, we see that

$$\tau_i = 1$$

and

$$\nu_i = \sum_{j=1}^{k} u_{i,j} E_{-z_i} g_j.$$

The expectation $E_{-z_i} g_j$ is equal to $E_q g_j = \mu_j$, because g_j is independent of z_i in the variational distribution. So, we conclude that the optimal update for $q_i(z_i)$ consists of only a hyperparameter update $\nu_i = \sum_{j=1}^{k} u_{i,j} \mu_j$.

Similarly, for g_j, we find that the optimal log variational density in our context is equal to

$$\sum_{i=1}^{n} u_{i,j} E_{-g_j} z_i g_j - \frac{1}{2} g_j^2 - \frac{1}{2}(\lambda_j + n^{-2})^q E_{-g_j} c g_j^2$$

up to a constant. Again, this is the log density of a normal distribution. The hyperparameter updates are given by

$$\mu_j = \frac{\sum_{i=1}^{n} u_{i,j} E_{-g_j} z_i}{1 + (\lambda_j + n^{-2})^q E_{-g_j} c}$$

and

$$\sigma_j^2 = \frac{1}{1 + (\lambda_j + n^{-2})^q E_{-g_j} c}.$$

Again, we can use the mean-field property to see that $E_{-g_j} z_i = E_q z_i$ and $E_{-g_j} c = E_q c$ and insert the values from Equations (5.5) and (5.9).

The optimal log variational density for c is equal to

$$(a + k/2 - 1)\log c - \left(b + \frac{1}{2} \sum_{j=1}^{k} (\lambda_j + n^{-2})^q E_{-c} g_j^2 \right) c,$$

up to a constant, implying that

$$\kappa = a + \frac{k}{2}$$

and

$$\theta = b + \frac{1}{2} \sum_{j=1}^{k} (\lambda_j + n^{-2})^q E_{-c} g_j^2,$$

where we have $E_{-c} g_j^2 = E_q g_j^2 = \mu_j^2 + \sigma_j^2$.

The above hyperparameter updates can be summarized in the following algorithm:

Algorithm 5.1 CAVI for classification with fixed truncation level k.

Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, fixed truncation level k, initial values for hyperparameters ν, μ, σ^2 and θ.
Output: ELBO and optimal variational approximation $q(c, g, z)$.

```
 1: repeat
 2:     for $i = 1, \dots, n$ do
 3:         Update
                $$\nu_i \leftarrow \sum_{j=1}^{k} u_{i,j}\mu_j.$$
 4:         Compute
                $$E_q z_i \leftarrow \begin{cases} \nu_i, & i \notin I^{\text{obs}}, \\ \nu_i + y_i \frac{\phi(\nu_i)}{\Phi(y_i \nu_i)}, & i \in I^{\text{obs}}. \end{cases}$$
 5:     end for
 6:     for $j = 1, \dots, k$ do
 7:         Update
                $$\mu_j \leftarrow \frac{\sum_{i=1}^{n} u_{i,j} E_q z_i}{1 + (\lambda_j + n^{-2})^q (a + k/2)/\theta}$$
            and
                $$\sigma_j^2 \leftarrow \frac{1}{1 + (\lambda_j + n^{-2})^q (a + k/2)/\theta}.$$
 8:     end for
 9:     Update
                $$\theta \leftarrow b + \frac{1}{2} \sum_{j=1}^{k} (\lambda_j + n^{-2})^q (\mu_j^2 + \sigma_j^2).$$
10:     Compute ELBO using Equation (5.4).
11: until convergence of ELBO.
```

5.4. STOCHASTIC VARIATIONAL INFERENCE

Coordinate ascent is not the only way to approximately maximize the ELBO. An alternative and more traditional method to maximize ELBO is gradient ascent (see e.g. Blei et al. (2017); Hoffman et al. (2013); Paisley et al. (2012); Ranganath et al. (2014) in the context of variational inference). We view the ELBO (5.4) as a function of the hyperparameters ν, μ, σ^2 and θ. The gradient ascent method updates the hyperparameters in each iteration in the direction of the gradient of the ELBO. If the step size is appropriate (cf. Wolfe (1969)), we find a local maximum.

5.4.1. STOCHASTIC OPTIMIZATION

A computational consideration in both the CAVI algorithm and the gradient ascent method, is the fact that there is a hyperparameter v_i associated to every vertex $i = 1, \dots, n$ of the graph and that the update of the hyperparameter μ_j uses all v_i's. If the graph is very large, this results in very slow iterations. The MCMC method in Hartog and Van Zanten (2019) (Chapter 3) and the empirical Bayes method in Chapter 4 have a similar feature. In our hierarchical model (5.2), we see that each data point (i, y_i) is associated to a latent variable z_i, whereas the parameters g and c influence all data points. For this reason, we make a distinction between the *local* hyperparameter ν and the *global* hyperparameters μ, σ^2 and θ (cf. Blei et al. (2017); Hoffman et al. (2013)). Following the approach of Robbins and Monro (1951), we replace the exact gradient with a noisy gradient that only takes one (or a few) data point into account per iteration instead of all data points to speed up iterations:

- Draw l uniformly at random from $\{1, \dots, n\}$.
- Update the local hyperparameter $\nu_l \leftarrow \sum_{j=1}^k u_{l,j}\mu_j$.
- Estimate $\sum_{i=1}^n u_{i,j} E_q z_i$ by $n u_{l,j} E_q z_l$, where

$$E_q z_l = \begin{cases} \nu_l, & l \notin I^{\text{obs}}, \\ \nu_l + y_l \frac{\phi(\nu_l)}{\Phi(y_l \nu_l)}, & l \in I^{\text{obs}}. \end{cases}$$

- Use the estimate $n u_{l,j} E_q z_l$ to update the global hyperparameters.

Note that $n u_{l,j} E_q z_l$ is an unbiased estimator for $\sum_{i=1}^n u_{i,j} E_q z_i$. A straightforward generalization is to sample a subset $B \subset \{1, \dots, n\}$ of fixed size $\#B$ instead of a single vertex. The estimate for $\sum_{i=1}^n u_{i,j} E_q z_i$ is in that case

$$\frac{n}{\#B} \sum_{i \in B} u_{i,j} E_q z_i. \tag{5.12}$$

The random subset B is also called a *minibatch* (Blei et al., 2017). In the last step, we use the estimate (5.12) to construct a noisy estimate of the gradient of the ELBO with respect to the global hyperparameters. This technique is called *stochastic optimization*, and the whole procedure *stochastic variational inference* (cf. Blei et al. (2017); Hoffman et al. (2013)).

The size of the minibatches is an additonal hyperparameter which has to be set. A small minibatch size increases the noise in the estimate (5.12), but speeds up each iteration of the resulting algorithm. A large minibatch size results in a more accurate estimate of the gradient, but increases the computational costs of each iteration. The optimal balance depends on the computational architecture (see, e.g. Bottou et al. (2018)).

5.4.2. GRADIENT OF THE ELBO

To construct a noisy estimate of the gradient of the ELBO with respect to the global hyperparameters, we can compute the partial derivatives of Equation (5.4).

$$\begin{aligned}
\frac{\partial \mathrm{ELBO}}{\partial \mu_j} &= \sum_{i=1}^{n} u_{i,j} E_q z_i - \mu_j - (\lambda_j + n^{-2})^q \frac{a + k/2}{\theta} \mu_j, \\
\frac{\partial \mathrm{ELBO}}{\partial \sigma_j^2} &= -\frac{1}{2} - \frac{1}{2} \frac{a + k/2}{\theta} (\lambda_j + n^{-2})^q + \frac{1}{2} \frac{1}{\sigma_j^2}, \\
\frac{\partial \mathrm{ELBO}}{\partial \theta} &= -\frac{a + k/2}{\theta} + \left(b + \frac{1}{2} \sum_{j=1}^{k} (\lambda_j + n^{-2})^q (\mu_j^2 + \sigma_j^2) \right) \frac{a + k/2}{\theta^2}.
\end{aligned}$$

We can adapt the above gradient to take the different roles of the hyperparameters μ, σ^2 and θ into account (cf. Amari (1998)). If the parameters where orthonormal, the direction of the gradient would be direction of the steepest increase in ELBO. However, due to the interplay of the hyperparameters via $q(c, g, z)$, the direction of the steepest increase is

$$I(\mu, \sigma^2, \theta)^{-1} \nabla \mathrm{ELBO},$$

where $I(\mu, \sigma^2, \theta)$ is the Fisher information matrix, defined by

$$I(\mu, \sigma^2, \theta) = E_q (\nabla \log q(c, g, z))(\nabla \log q(c, g, z))^T,$$

where the gradient is taken with respect to the global hyperparameters μ, σ^2 and θ (see, e.g. Amari (1998); Blei et al. (2017); Honkela et al. (2007)). This is called the *natural gradient* with respect to the global hyperparameters. Note that due to the mean-field factorization of $q(c, g, z)$, the factor corresponding to z does not play a role in this computation. Moreover, the independence between g and c and between g_j and g_l, for $l \neq j$, implies that the Fisher information matrix has zero entries on the blocks corresponding to those terms. Further computation shows that $I(\mu, \sigma^2, \theta)$ is a diagonal matrix with entries

- $\frac{1}{\sigma_j^2}$, corresponding to μ_j,
- $\frac{1}{2\sigma_j^4}$, corresponding to σ_j^2 and
- $\frac{a+k/2}{\theta^2}$, corresponding to θ.

All in all, after updating v using a minibatch B, we use the noisy natural gradient

$$\nabla_{\mu_j} = \sigma_j^2 \frac{n}{\#B} \sum_{i \in B} u_{i,j} E_q z_i - \sigma_j^2 \left(1 + (\lambda_j + n^{-2})^q \frac{a + k/2}{\theta} \right) \mu_j$$

to update μ_j for $j = 1, \ldots, k$,

$$\nabla_{\sigma_j} = \sigma_j^2 - \left(1 + (\lambda_j + n^{-2}) \frac{a + k/2}{\theta} \right) \sigma_j^4$$

to update σ_j^2 for $j = 1, \ldots, k$, and

$$\nabla_\theta = -\theta + b + \frac{1}{2} \sum_{j=1}^{k} (\lambda_j + n^{-2})^q (\mu_j^2 + \sigma_j^2)$$

to update θ. The updates consist of taking a step in the direction of the natural gradient. For example, in iteration t, the hyperparameter θ is updated to $\theta + \epsilon\nabla_\theta$. The step size $\epsilon = \epsilon_t$ can depend on the iteration t. If the sequence ϵ_t satisfies $\sum_{t=1}^{\infty} \epsilon_t = \infty$ and $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$, the resulting algorithm will find a local maximum (cf. Bottou (1998); Robbins and Monro (1951)). An example sequence that satisfies these conditions is $\epsilon_t = 1/t$. These steps can be summarized in the following algorithm:

Algorithm 5.2 SVI for classification with fixed truncation level k.

```
Input: Data $D = \{(i, y_i) : i \in I^{\text{obs}}\}$, fixed truncation level $k$, initial values for hyperparameters $\nu$, $\mu$, $\sigma^2$ and $\theta$, $t = 0$ and a sequence $\epsilon_t$.
Output: ELBO and optimal variational approximation $q(c, g, z)$.
 1: repeat
 2:     Draw a random subset $B$ of size $\#B$ from $\{1, \dots, n\}$.
 3:     for $i \in B$ do
 4:         Update
                $$\nu_i \leftarrow \sum_{j=1}^{k} u_{i,j}\mu_j.$$
 5:         Compute
                $$E_q z_i \leftarrow \begin{cases} \nu_i, & i \notin I^{\text{obs}}, \\ \nu_i + y_i \frac{\phi(\nu_i)}{\Phi(y_i \nu_i)}, & i \in I^{\text{obs}}. \end{cases}$$
 6:     end for
 7:     Compute the noisy natural gradient
                $$\nabla_\theta \leftarrow -\theta + b + \frac{1}{2}\sum_{j=1}^{k}(\lambda_j + n^{-2})^q(\mu_j^2 + \sigma_j^2).$$
 8:     for j=1, ..., k do
 9:         Compute the noisy natural gradient
                $$\nabla_{\mu_j} \leftarrow \sigma_j^2 \frac{n}{\#B}\sum_{i \in B} u_{i,j} E_q z_i - \sigma_j^2\left(1 + (\lambda_j + n^{-2})^q \frac{a + k/2}{\theta}\right)\mu_j,$$
                $$\nabla_{\sigma_j^2} \leftarrow \sigma_j^2 - \left(1 + (\lambda_j + n^{-2})\frac{a + k/2}{\theta}\right)\sigma_j^4.$$
10:         Update
                $$\mu_j \leftarrow \mu_j + \epsilon_t \nabla_{\mu_j},$$
                $$\sigma_j^2 \leftarrow \sigma_j^2 + \epsilon_t \nabla_{\sigma_j^2}.$$
11:     end for
12:     Update
                $$\theta \leftarrow \theta + \epsilon_t \nabla_\theta.$$
13:     Compute ELBO using Equation (5.4).
14: until convergence of ELBO.
```

Note that the computation of the ELBO using Equation (5.4) also involves a sum of $E_q \log p(z_i \mid g)$ and $E_q \log q_i(z_i)$ over all vertices $i = 1, \dots, n$. This would become the new bottleneck in the computation. As an alternative, we could use the average log predictive probability density on a small held-out dataset $D' \subset D$ to monitor the progress of the algorithm, as proposed in Blei et al. (2017). However, we prefer to use all the available data in our inference. In that case, a cheaper alternative is to only compute the ELBO every once every few, for example $n/\#B$, iterations.

5.5. SAMPLING

So far, we have described algorithms to find hyperparameters for an approximation to the posterior distribution $p(c, g, z \mid D)$ for a fixed truncation level k. However, we do not know the correct truncation level beforehand. As we have a conjugate prior for c, we treat it as an unknown parameter of the model that we estimate in the posterior. In contrast, we can view k as another hyperparameter of the ELBO. We choose to separate the optimization over the hyperparameters ν, μ, σ^2 and θ from the optimization over k, as k is a discrete parameter (as in Chapter 4). Traditional methods for the selection of discrete hyperparameters are manual search and grid search (see Bergstra and Bengio (2012)). The space for the hyperparameter k is one-dimensional, so we propose a grid search. As the computational costs of the optimization of the hyperparameters ν, μ, σ^2 and θ increase with k, we propose a bottom-up search. We start with a low value for k and find the corresponding optimal hyperparameters in terms of the ELBO. Then, we increase k, so we include one more eigenvector of the Laplacian matrix to the series expansion of f. At some point, adding additional eigenvalues as explanatory variables does not increase the ELBO significantly. This is where we break the search. This method has two potential disadvantages. First, it exhaustively searches through all possible values of k, although it does so in the computationally optimal order. Second, it might be that the ELBO has very small increments for low values of k followed by a large increment later on. Depending on the stopping rule for the bottom-up search, we might never reach that large increment (see for example Section 4.5.2). The proposed bottom-up search is summarized in Algorithm 5.3.

Algorithm 5.3 Bottom-up search for k^*.

```
Input: Initial $k$.
Output: Optimal truncation level $k^*$.
1: repeat
2:     Compute the hyperparameters and ELBO using Algorithm 5.1 or 5.2.
3:     Update $k \leftarrow k + 1$.
4: until convergence.
```

After we have found the optimal truncation level k, we can sample from the approximate posterior $q(c, z, g)$ using the hyperparameters that we found with Algorithm 5.1 or 5.2. Because of the factorization (5.3), we see that the marginal for g_j is simply $q_j(g_j) \sim N(\mu_j, \sigma_j^2)$ and that it is independent of z, c and g_l for $l \neq j$. This makes sampling from the approximate posterior g straightforward.

5.6. NUMERICAL EXPERIMENTS

5.6.1. MINIBATCH SIZE

To illustrate the impact of the minibatch size to the maximization of the ELBO, we consider a simple example on the path graph of 1000 vertices. We define a function on the graph by

$$f(i) = \frac{\sin(12(i/n + 0.2))}{i/n + 0.2}.$$

We assign labels to the vertices according to probabilities $P(y_i = 1) = 1 - P(y_i = -1) = \Phi(f_0(i))$, where Φ is the distribution function of the standard normal distribution. We remove 20% of the labels.

We fix k at 25 for now, as we are only interested in the impact of choosing different minibatch sizes. We run Algorithm (5.2) with the same initial value for minibatch sizes 1, 2, 5, 10, 20, 50, 100, 200, 500 and 1000. A minibatch of size 1000 consists of all the vertices. As the computational cost of Algorithm (5.2) is linear in the minibatch size, we allow more iterations for smaller minibatch sizes. More specifically, we allow for $10^6/\#B$ iterations for minibatch size $\#B$. The other hyperparameters we set are $q = 1.5$ and $\epsilon_t = 1/(t+10)$. The result in Figure 5.1 suggests that smaller minibatches find the (local) maximum of the ELBO faster. We also observe that the true underlying function is smoother than the approximate posterior. This suggests that $k = 25$ is too high. Indeed, a choice of $k = 11$, results in a better fit to the true function and also a higher ELBO (see Figure 5.2). We observe the same behavior of the ELBO as with $k = 25$.

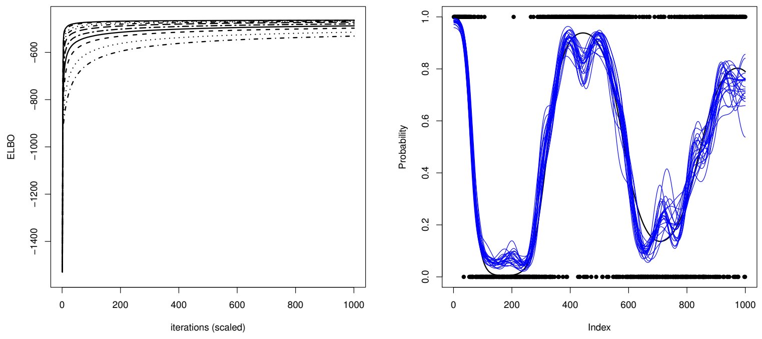


Figure 5.1: Example with $k = 25$. Left: ELBO value as a function of (scaled) iteration. The iteration is scaled as $10^{-3}\#Bt$, reflecting approximately how many times we have looped through the vertices. The minibatch sizes are 1, 2, 5, 10, 20, 50, 100, 200, 500 and 1000 from top to bottom. Right: The observations (black dots), the true underlying function (black line), the approximation of the posterior mean (thick blue line) and a sample of 20 draws from the approximate posterior (blue lines).

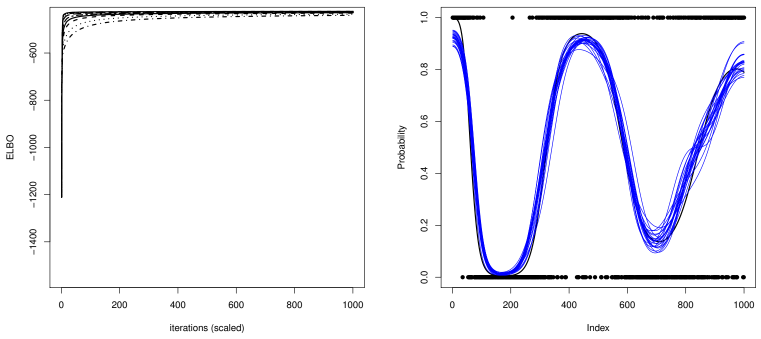


Figure 5.2: The same as Figure 5.1, but with $k = 11$.

5.6.2. CHANGING OBJECT IN A NOISY ENVIRONMENT

In this example, we apply our method to a large graph constructed in the following way. As a ground truth, we use the following animation of a morphing object. The animation consists of 60 frames of 60×60 pixels. The labels on each pixel indicate whether the pixel is of the object or not. We flip 10% of the labels at random. To convert this problem into a graph problem, we connect neighboring pixels in each frame and with the corresponding pixels in the previous and next frame, resulting in a $60 \times 60 \times 60$ grid graph on a total of $n = 216000$ nodes. We can explicitly compute the eigenvalues using the known eigenvalues of the path graph using Theorem 3.5 in Mohar (1991). The corresponding eigenvectors are given by the tensor products of the eigenvectors of the path graph and can also be computed explicitly, see Chapter 1 for more details. We estimate the soft label function using Algorithm 5.2. The estimated soft label function is shown in Figure 5.4. We can truncate it at 0.5 to estimate the shape of the object in each frame. The estimated objects are shown in Figure 5.5.

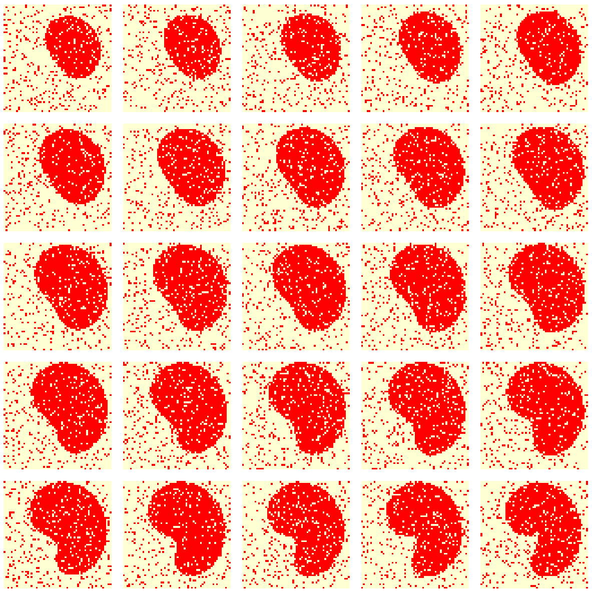


Figure 5.3: 25 frames of the true underlying morphing object with noise.

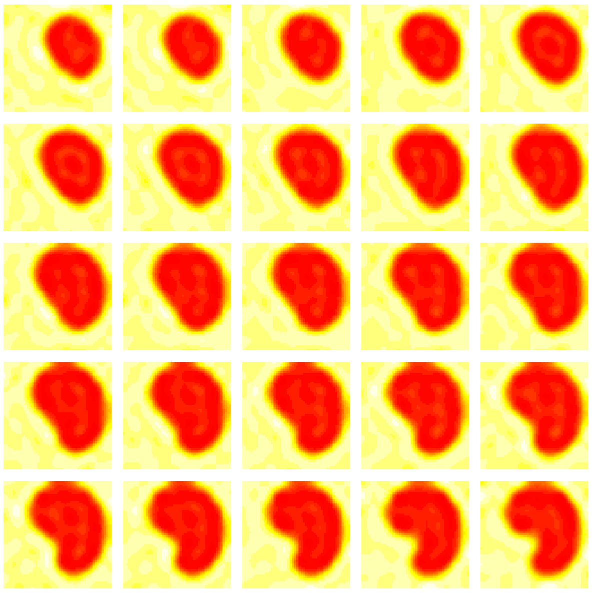


Figure 5.4: 25 frames of the estimated soft label function.

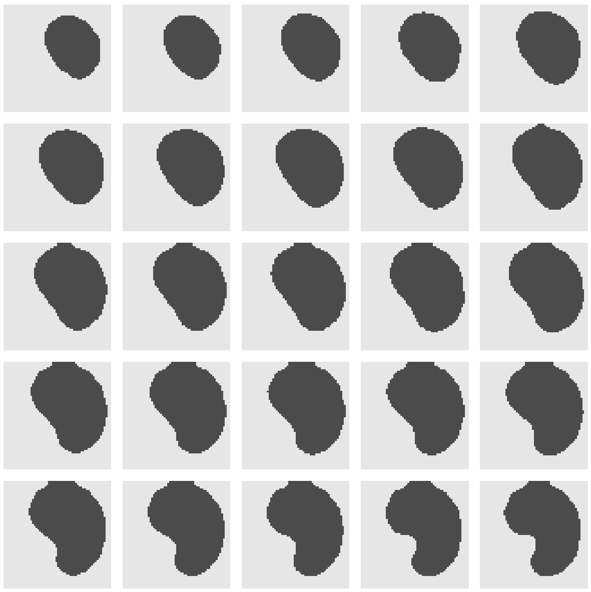


Figure 5.5: 25 frames of the estimated labels, using truncation at 0.5.

5.7. CONCLUDING REMARKS

We have described a nonparametric Bayes procedure to perform binary classification on graphs. We have considered a variational Bayes approach using a prior that can be represented as a scaled, truncated series of Gaussians. The proposed method consists of approximating the posterior distribution with a family of variational distributions and finding the optimal hyperparameters. These hyperparameter are chosen, such that the evidence lower bound is maximized. We have described coordinate ascent variational inference and stochastic variational inference. An advantage of the latter method is that the precise gradient is replaced by a noisy estimate of the gradient, which can be much cheaper to compute. The noisy estimate of the gradient is based on a subset of the observed labels, a minibatch, instead of the entire data. The minibatch size is an additional hyperparameter which has to be set. Our numerical example suggests that small minibatch sizes are preferred over large minibatch sizes. Additionally, we have to determine a sequence ϵ_t. Another disadvantage of variational inference is that the variational distribution might not be a good approximation to the true posterior distribution. Recently, some theoretical results are available about convergence of variational posteriors (e.g. Zhang and Gao (2018)), but this remains a difficult problem in general. We have observed good results using stochastic variational inference in a simulated example with a graph of over 200000 vertices.

DANKWOORD

Allereerst bedank ik Harry voor het mogelijk maken van het onderzoek dat tot dit proefschrift heeft geleid, jouw bijdrage hieraan en sturing. Van jou heb ik geleerd dat een succesvolle onderzoeker alles van alle onderwerpen moet weten. Jij bent hier een goed voorbeeld van en ik heb een hoop van jou geleerd. Ik ben bijzonder dankbaar voor het vertrouwen in mij om het laatste deel van het traject in Londen af te maken en de kans die je mij hebt gegeven om mijn onderzoek te combineren met mijn baan bij EY.

Bij EY hebben Wimjan, Floris, Philippe, Niels en Rens er voor gezorgd dat deze combinatie zo soepel mogelijk verliep. Bij elke terugkomst na een periode van focus op mijn onderzoek kon ik altijd rekenen op een warm onthaal en door mee te blijven doen aan de wintersport en andere activiteiten heb ik mij altijd deel gevoeld van het team.

Het andere team waar ik deel van was, was mijn onderzoeksgroep bestaande uit Jan, Alice, Botond en Moritz. Het was fijn om met jullie te werken en om over andere dingen dan werk te praten. Ik heb veel aan jullie ervaring gehad. Jullie zijn stuk voor stuk inspirerende onderzoekers die de bredere onderzoeksgemeenschap interessant houden.

Voor verdere inhoudelijke adviezen en feedback op mijn manuscript kon ik uiteraard rekenen op Teun en Jacob. Hoewel dit proefschrift slechts wisselgeld is in verhouding tot het grote goud dat jullie mij waard zijn, is het toch fijn om ook hier jullie steun te genieten.

Ik ben altijd gesteund door mijn ouders Wim en Jutta. Jullie hebben mij vanaf nul gestimuleerd om te leren. Door jullie ben ik slim. Vroeger kon ik trots mijn tekeningen laten zien en nu moeten jullie het doen met computergegenereede plaatjes die blijkbaar iets betekenen. Gelukkig weten jullie altijd wat echt belangrijk is en dat leer ik nog steeds van jullie.

Ook Jelmer weet wat echt is. Je hebt mij veel geholpen met het afronden van mijn proefschrift en ik hoop ook jou trots te kunnen maken met dit werk.

Lieve Merel, ontzettend bedankt voor alle liefde, hulp en wijsheid. Jij was er bij alle fases van het onderzoek bij en je was overal van onschatbare waarde. Ik kijk uit naar alle volgende fases in ons leven. Jij betekent alles voor mij.

LIST OF PUBLICATIONS

1. J. Hartog and J.H. van Zanten, *Nonparametric Bayesian label prediction on a graph*, Computational Statistics & Data Analysis **120**, 111-131 (2018).

2. J. Hartog and J.H. van Zanten, *Nonparametric Bayesian label prediction on a large graph using truncated Laplacian regularization*, Communications in Statistics - Simulation and Computation (2019).

Parts of Chapter 2 are published in item 1 (Hartog and Van Zanten, 2018). J. Hartog's contribution includes execution of the research, numerical experiments and draft of the article. J.H. van Zanten's contribution includes discussion and development of the concept, guidance and review of the article.

Parts of Chapter 3 are published in item 2 (Hartog and Van Zanten, 2019). J. Hartog's contribution includes execution of the research, numerical experiments and draft of the article. J.H. van Zanten's contribution includes discussion and development of the concept, guidance and review of the article.

REFERENCES

J. H. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993.

D. J. Aldous and J. Shun. Connected spatial networks over random points and a route-length statistic. *Statistical Science*, 25(3):275–288, 2010.

S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276, 1998.

R. K. Ando and T. Zhang. Learning on graph with Laplacian regularization. In *Advances in neural information processing systems*, pages 25–32, 2007.

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning theory*, volume 3120 of *Lecture notes in computational science*, pages 624–638. Springer, Berlin, 2004.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7:2399–2434, 2006.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

A. L. Bertozzi, X. Luo, A. M. Stuart, and K. C. Zygalakis. Uncertainty quantification in graph-based classification of high dimensional data. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):568–595, 2018.

P. J. Bickel, C. Chen, J. Kwon, J. Rice, E. Van Zwet, and P. Varaiya. Measuring traffic. *Statistical Science*, pages 581–597, 2007.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

M. Blair, R. Armstrong, and M. Murphy. *The 360 degree brand in Asia: creating more effective marketing communications*. Wiley, New York, 2003.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

L. Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998. revised, oct 2012.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov chain Monte Carlo*. CRC press, Boca Raton, FL, 2011.

N. Chopin. Fast simulation of truncated Gaussian distributions. *Statistics and Computing*, 21(2):275–288, 2011.

N. Choudhuri, S. Ghosal, and A. Roy. Nonparametric binary regression using a Gaussian process prior. *Statistical Methodology*, 4(2):227–243, 2007.

D. Cvetkovic, S. Simic, and P. Rowlinson. *An introduction to the theory of graph spectra*. Cambridge University Press, Cambridge, 2009.

A. Damianou, M. K. Titsias, and N. D. Lawrence. Variational gaussian process dynamical systems. In *Advances in Neural Information Processing Systems*, volume 24, pages 2510–2518, 2011.

L. Devroye. *Non-uniform random variate generation*. Springer-Verlag, New York, 1986.

R. Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1997.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer Series in Statistics. Springer-Verlag, New York, 2001.

A. Gelman, H. S. Stern, J. B. Carlin, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. Statistical Science Series. CRC Press, Boca Raton, FL, third edition, 2014.

S. Ghosal and A. Van der Vaart. *Fundamentals of nonparametric Bayesian inference*, volume 44 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2017.

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

J. Hartog and J. H. Van Zanten. Nonparametric Bayesian label prediction on a graph. *Computational Statistics & Data Analysis*, 120:111–131, 2018.

J. Hartog and J. H. Van Zanten. Nonparametric bayesian label prediction on a large graph using truncated laplacian regularization. *Communications in Statistics - Simulation and Computation*, 2019.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

A. Honkela, M. Tornio, T. Raiko, and J. Karhunen. Natural conjugate gradient in variational inference. In *Neural Information Processing*, pages 305–314, 2007.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

A. Kirichenko and J. H. Van Zanten. Estimating a smooth function on a large graph by Bayesian Laplacian regularisation. *Electronic Journal of Statistics*, 11(1):891–915, 2017.

E. D. Kolaczyk. *Statistical analysis of network data.* Springer Series in Statistics. Springer, New York, 2009.

H. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications.* Springer, New York, 1997.

F. Liang, K. Mao, M. Liao, S. Mukherjee, and M. West. Nonparametric Bayesian kernel models. Technical report, Department of Statistical Science, Duke University, 2007.

X. Liu, D. Zhao, J. Zhou, W. Gao, and H. Sun. Image interpolation via graph-based Bayesian label propagation. *IEEE Transactions on Image Processing*, 23(3):1084–1096, 2014.

N. Madras. *Lectures on Monte Carlo methods*, volume 16 of *Fields Institute Monographs.* American Mathematical Society, Providence, RI, 2002.

B. Mohar. The Laplacian spectrum of graphs. In *Graph theory, combinatorics, and applications*, volume 2 of *Wiley-Interscience Publication*, pages 871–898. Wiley, New York, 1991.

N. Nariai, E. D. Kolaczyk, and S. Kasif. Probabilistic protein function prediction from heterogeneous genome-wide data. *Plos one*, 2(3):e337, 2007.

J. Paisley, D. M. Blei, and M. I. Jordan. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*, volume 29, pages 1367–1374, 2012.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press, 2007.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2018. URL `https://www.R-project.org/`.

R. Ranganath, S. Gerrish, and D. M. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes in machine learning.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

C. P. Robert. *The Bayesian choice.* Springer Texts in Statistics. Springer, New York, 2007.

V. Sadhanala, Y.-X. Wang, and R. J. Tibshirani. Total variation classes beyond 1d: Minimax rates, and the limitations of linear smoothers. In *Advances in Neural Information Processing Systems*, pages 3513–3521, 2016.

R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Molecular systems biology*, 3(1):88, 2007.

V. Sindhwani, W. Chu, and S. S. Keerthi. Semi-supervised Gaussian process classifiers. In *International Joint Conference on Artificial Intelligence*, pages 1059–1064, 2007.

K. T. Smith. Digital marketing strategies that millennials find appealing, motivating, or just annoying. *Journal of Strategic Marketing*, 19(6):489–499, 2011.

A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, New York, 2003.

L. Tierney. A note on Metropolis-Hastings kernels for general state spaces. *The Annals of Applied Probability*, 8(1):1–9, 1998.

M. K. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 844–851, 2010.

A. B. Tsybakov. *Introduction to nonparametric estimation*. Springer Series in Statistics. Springer, New York, 2009.

F. Van der Meulen, M. Schauer, and J. H. Van Zanten. Reversible jump MCMC for non-parametric drift estimation for diffusion processes. *Computational Statistics & Data Analysis*, 71:615–632, 2014.

J. Van Waaij and J. H. Van Zanten. Full adaptation to smoothness using randomly truncated series priors with Gaussian coefficients and inverse gamma scaling. *Statistics & Probability Letters*, 123:93–99, 2017.

R. Waagepetersen and D. Sorensen. A tutorial on reversible jump mcmc with a view toward applications in qtl-mapping. *International Statistical Review*, 69(1):49–61, 2001.

L. Wasserman. *All of nonparametric statistics*. Springer Texts in Statistics. Springer, New York, 2006.

D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393 (6684):440, 1998.

D. B. West. *Introduction to graph theory*. Prentice Hall, Upper Saddle River, NJ, 1996.

P. Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

F. Zhang and C. Gao. Convergence rates of variational posterior distributions. *arXiv preprint arXiv:1712.02519*, 2018.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 14(1):185–205, 2005.